Министерство науки и высшего образования РФ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»
Институт математики, физики, информатики и технологий
Кафедра информатики, информационных технологий и
методики обучения информатике

ОРГАНИЗАЦИЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ УЧАЩИХСЯ 8-9 КЛАССОВ В ПРОЦЕССЕ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ PYTHON

Выпускная квалификационная работа
бакалавра по направлению подготовки
44.03.01 – Педагогическое образование
Профиль «Информатика»

Работа допущена к защите
«____»_____ 2019 г.
Зав. кафедрой _____

Исполнитель: студентка группы ИНФ-1501
Института математики,
физики, информатики и
технологий
Омарова Г.Р.
_____

Руководитель: старший преподаватель
кафедры ИИТиМОИ
Шимов И.В.
_____

Екатеринбург – 2019

Реферат

Омарова Г.Р. ОРГАНИЗАЦИЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ УЧАЩИХСЯ 8-9 КЛАССОВ В ПРОЦЕССЕ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ PYTHON, выпускная квалификационная работа: стр. 70, рис. 14, табл. 3, библ. 37 назв.

Ключевые слова: методика обучения, языки программирования, Python, самостоятельная работа, базовый курс информатики, алгоритмизация и программирование.

Объект исследования: процесс обучения программированию с использованием современных языков программирования в базовом курсе информатики.

Цель исследования: разработать методические рекомендации по организации самостоятельной работы учащихся в средней школе при обучении программированию с использованием современных языков программирования.

Работа посвящена разработке методики организации самостоятельной работы учащихся при изучении алгоритмизации и программирования с использованием современных языков программирования в базовом курсе информатики. Обсуждаются результаты сравнительного анализа языков программирования и различных УМК по информатике. Рассматриваются средства и способы организации самостоятельной работы учащихся. Кратко описываются основные характеристики онлайн-платформы Stepic и исследуются возможности их применения при организации самостоятельной работы про изучении языка программирования Python. Представлены методические рекомендации для учителя по организации курса для самостоятельной работы учащихся при изучении алгоритмизации и программирования.

Оглавление

Введение

Информационные технологии – неотъемлемая часть современного общества. Сложно придумать пример современной профессии, в которой не пригодилось бы умение работать с компьютером. Даже если профессия не вязана с IT технологиями, умение работать в специализированных программах или пользоваться хотя бы основными офисными приложениями необходимо и врачам, и учителям, и инженерам.

В ряде профессий нелишним является и навык программирования: это облегчает понимание работы системы и дает возможности повысить за счет этого производительность. Многие современные специализированные программы включают в себя элементы программирования. Даже стандартный пакет офисных программ, с которым знакомится каждый школьник и которым пользуется практически каждый человек, взаимодействующий с компьютерами, при использовании программирования можно значительно расширить возможности программы и области ее применения.

Программирование является важным элементом в образовании современных школьников, так как в первую очередь развивает алгоритмическое мышление, умение планировать свою деятельность, разбивать сложные задачи на более простые подзадачи и т.д. Крайне важно при изучении данной темы использовать современные языки программирования для повышения мотивации учащихся к изучению программирования и повышения актуальности полученных знаний. К сожалению, в современном курсе информатики на алгоритмизацию и программирование отводится недостаточно часов для более качественного изучения даже одного языка программирования и его средств, поэтому часть работы учащимся необходимо проделывать самостоятельно. По-нашему мнению, у учащихся 8-9 классов уровень навыка самостоятельной работы не соответствует необходимому, поэтому учителю нужно грамотно организовать самостоятельную работу учащихся, дать им возможность развития самостоятельности в их учебной и познавательной деятельности.

В вопросе организации самостоятельной работы учащихся в рамках изучения программирования нет конкретных педагогических технологий, соответствующих действующему федеральному образовательному стандарту. Поэтому организация самостоятельной работы учащихся в процессе обучения программированию с использованием современного языка программирования является актуальной темой для исследования.

Целью данной работы является разработка методических рекомендации по организации самостоятельной работы учащихся в средней школе при обучении программированию с использованием современных языков программирования.

Объект исследования: процесс обучения программированию с использованием современных языков программирования в базовом курсе информатики.

Предмет: современные средства и технологии для организации самостоятельной учебно-познавательной деятельности учащихся при обучении программированию в базовом курсе информатики.

Задачи:

1. Проанализировать нормативные документы основного общего образования на наличие требований к изучаемым языкам программирования и самостоятельной работе учащихся.
2. Провести сравнительный анализ распространенных языков программирования на степень возможности обучения на них.
3. Рассмотреть виды и способы организации самостоятельной работы, провести сравнительный анализ средств для организации самостоятельной работы учащихся.
4. Провести сравнительный анализ авторских учебных программ по информатике и ИКТ.
5. Разработать методические рекомендации по организации самостоятельной работы учащихся при обучении

программированию с использованием одного из современных языков программирования и провести апробацию разработок.

ГЛАВА 1. Теоретические основы

1.1. Самостоятельная работа учащихся

Одним из основных законодательных документов в области основного общего образования является Федеральный государственный образовательный стандарт основного общего образования, регламентирующий требования к образовательному процессу, учебной и внеурочной деятельности, требования к усвоению образовательной программы и развитию учащихся.

1.1.1. Требования ФГОС к самостоятельной работе учащихся

Методологической основой Федерального государственного стандарта основного общего образования в Российской Федерации является деятельностный подход в обучении, который обеспечивает:

- формирование готовности обучающихся к саморазвитию и непрерывному образованию;
- активную учебно-познавательную деятельность обучающихся;
- построение образовательного процесса с учётом индивидуальных, возрастных, психологических, физиологических особенностей и здоровья обучающихся.

В соответствии требованиям стандарта учителя должны обеспечить самостоятельную деятельность учащихся в рамках изучения предмета, возможность построения собственного образовательного маршрута учащихся, способствовать развитию самоконтроля и самоорганизации учащихся.

Согласно ФГОС учебная программа должна обеспечивать:

- развитие у обучающихся способности к самопознанию, саморазвитию и самоопределению;
- формирование умений самостоятельного планирования и осуществления учебной деятельности и организации учебного сотрудничества с педагогами и сверстниками, построения индивидуального образовательного маршрута;

- создание условий для интеграции урочных и внеурочных форм учебно-исследовательской и проектной деятельности обучающихся, а также их самостоятельной работы по подготовке и защите индивидуальных проектов;
- возможность практического использования приобретенных обучающимися коммуникативных навыков, навыков целеполагания, планирования и самоконтроля;
- подготовку к осознанному выбору дальнейшего образования и профессиональной деятельности. [27]

В связи с развитием самостоятельности обучающихся, ФГОС регламентирует процентное соотношение обязательной части образовательной программы среднего общего образования и самостоятельной работы учащихся. Обязательная часть образовательной программы среднего общего образования составляет 60%; часть, формируемая участниками образовательных отношений, – 40% от общего объема образовательной программы среднего общего образования. [27]

1.1.2. Сущность самостоятельной работы учащегося

П.И. Пидкасистый считает, что самостоятельная работа – это не форма организации учебных знаний и не метод обучения. Ее правомерно рассматривать скорее, как средство организации и выполнения учащимися определенной детальности в соответствии с поставленной целью. [25]

Во-первых, самостоятельная работа школьника – следствие правильно организованной учебной деятельности на уроке, что мотивирует самостоятельное ее расширение, углубление и продолжение в свободное время.

Во-вторых, самостоятельная работа – более широкое понятие, чем домашняя работа (выполнение заданий, данных учителем в классе на дом для подготовки к следующему уроку). Самостоятельная работа может включать внеурочную, задаваемую в той или иной форме учителем работу

обучающегося. Но в целом это параллельно существующая занятость учащегося. [16]

Как дидактическое явление, самостоятельная работа представляет собой, с одной стороны, учебное задание, т.е. то, что должен выполнить ученик, объект его деятельности, с другой – форму проявления соответствующей деятельности памяти, мышления, творческого воображения при выполнении учеником учебного задания, которое, в конечном счете, приводит школьника либо к получению совершенно нового, ранее неизвестного ему знания, либо к углублению и расширению сферы действия уже полученных знаний. [18]

Существенными особенностями, которые характеризуют самостоятельность учащегося в познавательном процессе являются: умение работать целенаправленно и по плану, выбирать наиболее рациональные приемы учебного труда, правильно рассчитывать свои силы и учитывать результаты собственной деятельности. [19]

«Самостоятельность» – невероятно многозначное слово, поэтому необходимо договориться, в каком именно значении мы его используем. В общем виде, т. е. в равной мере справедливом и для первоклассников, и для десятиклассников, слово «самостоятельный» используется в трех значениях:

- независимый, вольный, свободный, неподчиненный;
- действующий по собственной инициативе (от лат. initium – начало), сам себя побуждающий к началу какого-либо дела;
- совершаемый без чьей-либо помощи.

Иными словами, самостоятельность – это не только способность ребенка обходиться без помощи взрослого, но и способность запрашивать и получать необходимую помощь по собственной инициативе, и способность критично, независимо оценивать качество помощи, предлагаемой тем или иным источником (авторитетным взрослым, учебником и пр.). [37]

В своей работе мы будем понимать под самостоятельной работой – специально организованную деятельность, направленную на применение

полученных знаний и умений, поиск новых знаний, осуществляемую учащимися во внеурочное время без непосредственного контроля со стороны учителя.

Необходимость управления самостоятельной познавательной деятельностью учащегося следует из структуры педагогической системы. Компонентами педагогической системы являются цели, субъекты, реализующие эти цели, деятельность, отношения, возникающие между её участниками и объединяющее их управление, обеспечивающие единство системы. Утрата любого компонента ведёт к разрушению системы в целом. Учащиеся испытывают потребность в педагогическом руководстве в силу несовершенства их опыта самостоятельной познавательной деятельности. Даже хорошо подготовленным ученикам нужна помощь или консультация учителя, хотя не так часто, как остальным.

В процессе управления самостоятельной деятельностью преподаватель принимает прямое (затем косвенное) участие в организации процесса обучения. При этом целесообразно следовать ряду принципов управления:

- дифференцированный подход к учащимся с соблюдением посильности учебных заданий;
- планомерное возрастание интеллектуальных нагрузок и последовательный переход к более неточным и неполным указаниям по выполнению самостоятельной работы;
- постепенное отдаление учителя и занятие им позиции пассивного наблюдателя за процессом;
- переход от контроля учителя к самоконтролю. [18]

Благодаря усилиям дидактов и методистов, в теории обучения сформировались основные требования к проведению самостоятельных работ учащихся:

- соответствие содержания самостоятельных работ требованиям учебных программ;
- посильность самостоятельных работ для учащихся;

- соблюдение принципа сознательности при их выполнении;
- организация самостоятельных работ в определенной системе;
- подготовка учащихся к выполнению самостоятельных работ – точное, четкое, немногословное инструктирование учащихся о целях и задачах работы;
- вооружение их необходимыми техническими и организационными навыками для ее выполнения;
- постановка перед учащимися такой задачи, разрешение которой потребовало бы от них умственных усилий;
- соблюдение дозировки времени, отведенного на выполнение самостоятельного задания);
- непосредственное наблюдение учителя за ходом выполнения учащимися самостоятельной работы и оказание им необходимой помощи при возникновении затруднений;
- обязательная проверка выполнения учащимися самостоятельных работ. [11]

1.1.3. Виды самостоятельной работы

Основными видами классификации самостоятельной работы, выделяемые дидактами, являются:

- классификация по степени самостоятельности ученика:
    - работы по подражанию;
    - тренировочные работы;
    - упражнения;
    - работы творческого характера;
    - исследовательские работы и др.;
- классификация самостоятельных работ по дидактическому назначению самостоятельные работы:
    - для получения новых знания;
    - для применения знаний;

- для повторения и проверки знаний, умений и навыков.

Существует так же ряд авторских классификаций.

В.А. Добромыслов в целях развития у учащихся наблюдательности и мышления выделяет следующие виды самостоятельных работ:

1. Самостоятельные работы по наблюдению фактов и констатированию наблюдаемого.
2. По сопоставлению, сравнению и установлению связей между явлениями.
3. По выделению в событии, явлении наиболее существенного, важного с какой-нибудь точки зрения.
4. По определению типичного в ряде явлений, обобщению и обоснованию выводов.

В основу классификации В.П. Стрезикозина автор заложил источник знания и метод обучения. Исходя из этого в учебном процессе выделяют следующие виды самостоятельных работ:

1. Работа с учебником и учебной книгой.
2. Работа со справочной литературой.
3. Решение и сопоставление задач.
4. Учебные упражнения обычные и в тетрадях с печатной основой.
5. Сочинения и описания.
6. Наблюдения и лабораторные работы.
7. Работы-задания, связанные с использованием иллюстраций, карт, схем, графиков и раздаточного материала.
8. Графические работы.

Б.П. Есипов в своей работе «Самостоятельная работа учащихся на уроке» классифицирует самостоятельную работу по звеньям учебного процесса в соответствии с преобладанием в них основной дидактической цели:

1. Самостоятельные работы, применяемые с целью получения новых знаний.

2. Самостоятельные работы, применяемые на основе приобретенных знаний.
3. Самостоятельные работы, применяемые в целях повторения и проверки знаний, умений, навыков учащихся.

И.И. Малкин считает, что каждый тип и вид самостоятельной работы одновременно и определяет характер познавательной деятельности ученика и сам определяется ее структурой. Отсюда принцип классификации – степень самостоятельности и творчества учащихся при выполнении работы. Учитывая это обстоятельство, Малкин предлагает следующую классификацию:

1. Самостоятельные работы репродуктивного типа:
    a. Воспроизводительные.
    b. Тренировочные.
    c. Обзорные.
    d. Проверочные.
2. Самостоятельные работы творческого типа:
    a. Подготовительные.
    b. Констатирующие.
    c. Экспериментально-поисковые.
    d. Логически-поисковые.
3. Самостоятельные работ творческого типа:
    a. Художественно-образные.
    b. Научно-творческие.
    c. Конструктивно-технические.
4. Самостоятельные работы познавательно-практического типа:
    a. Учебно-практические.
    b. Общественно-практические. [25]

О.В. Петунин в своих работах выделяет 4 уровня познавательной самостоятельности. Под уровнем познавательной самостоятельности понимается совокупность ведущих (опорных) знаний, умений, навыков,

способов деятельности, которыми владеет обучающийся, и которые создают возможность их дальнейшего совершенствования.

Первый уровень – воспроизводящая самостоятельность – характеризуется тем, что обучающийся имеет опорные знания по предмету, соответствующие низкой ступени усвоения, когда учащиеся свидетельствуют, что раньше эти знания получали, но воспроизвести самостоятельно их не могли. Знания обучаемого на этом уровне ограничены наиболее общими представлениями об объекте изучения, наиболее общими чертами его облика. На этом уровне обучаемые неустойчиво владеют простейшими актами познавательной деятельности: анализа, сравнения, сопоставления, механически используют усвоенные знания и приемы познавательных действий в выполнении упражнений по образцу, испытывают затруднения при переносе знаний и действий в аналогичные условия с изменением 1–2х параметров. Этот уровень характеризуется отсутствием у обучающегося умений владеть более сложными формами мыслительных операций: абстрагирования, обобщения и др.

Второй уровень – реконструктивно-вариативная самостоятельность – характеризуется наличием у обучаемого опорных знаний, которые он может воспроизвести с помощью наводящих вопросов; умением переноса усвоенных знаний и способов деятельности в ситуации с изменением 2–3х параметров. Знания на этом уровне предполагают овладение основными понятиями, правилами, законами предмета настолько, что оно позволяет школьнику или студенту осуществлять словесное описание явлений, фактов, получаемых опытным путем, анализировать различные действия на основе применения усвоенных правил и законов. Этот уровень познавательной самостоятельности предполагает овладение обучающимися простейшими мыслительными операциями: анализа, синтеза, сравнения и частично другими.

Третий уровень – частичнопоисковая самостоятельность – характеризуется наличием опорных знаний по предмету, которые учащийся может воспроизвести самостоятельно, «без подсказок». Знания обучаемого на

этом уровне позволяют ему оперировать фактами, получаемых путем логических рассуждений, применять усвоенную информацию для решения задач и получения субъективно новой информации. Познавательная самостоятельность данного уровня характеризуется тем, что обучаемый может произвести расчленение сложного целого на части, выделение свойств, связей, отношений частей, главных и второстепенных признаков предметов и явлений, может систематизировать факты на основе установленных связей, но испытывает затруднения при раскрытии сущности новых понятий путем абстрагирования, обобщения, при использовании усвоенных знаний и навыков в совершенно новых ситуациях.

Четвертый уровень – творческая самостоятельность – характеризуется наличием более широкого и углубленного круга опорных знаний по предмету, которые обучаемый может самостоятельно актуализировать; умением найти новый подход в решении задачи и осуществить его. Этот уровень предполагает способность трансформировать исходные сведения настолько, что ему становятся посильны задачи любого класса. Деятельность обучаемого на этом уровне приобретает поисковый характер

Таким образом самостоятельная работа учащихся является обязательным компонентом в современном образовании. Необходимость развития навыков самостоятельной работы и использование данного вида организации учебной деятельности регламентировано ФГОС. Исходя из представленных мнений, самостоятельная работа учащихся – единая система, в которой важным является взаимодействие работы на уроке, в том числе самостоятельной работы учащегося на уроке, мотивацию учащегося к расширению и углублению полученных знаний в данной предметной области и его целенаправленную самостоятельную деятельность, организованную и контролируемую учителем. При этом со временем роль учителя во внеурочной самостоятельной работе должна уменьшаться и увеличиваться количество самоконтроля учащегося. Самостоятельная работа учит школьника не только ставить цели и искать пути их достижения, но и выбирать наиболее

оптимальные для достижения цели средства и приемы, рассчитывать свои силы и учитывать предыдущие результаты своей деятельности. Обязательным элементом в системе самостоятельной работы учащихся являются консультации с учителем, возможность получения помощи и наставлений в решении проблемы. Данный аспект самостоятельной работы развивает не только умение сотрудничества с другими людьми, но и планирование своей деятельности, формулирование вопросов.

В рамках изучения программирования оптимальными видами самостоятельной работы для организации является решение задач по составлению программы. Данное задание можно отнести к нескольким пунктам представленных выше классификаций:

- упражнения по решению задач, репродуктивные тренировочные задачи: решение типовых задач;
- задания, ориентированные на применение практических знаний: составление программы требует непосредственного использование полученных теоретических знаний и способствует лучшему запоминанию, так же подобные задачи повышают мотивацию, т.к. учащийся сразу видит область применения и практическую пользу изученного материала;
- творческие работы и логически-поисковые задачи: с повышением уровня сложности задачи возрастает роль творческой составляющей.

Решение алгоритмических задач различного уровня сложности позволяет оценить не только уровень предметных навыков и умений, но и уровень развитости познавательной самостоятельности учащихся.

1.2. Современные языки программирования и требования к выбору языка

1.2.1. Требования ФГОС к языку программирования

Согласно ФГОС изучение предметной области "Математика и информатика" должно обеспечить сформированность:

- основ логического, алгоритмического и математического мышления;
- умений применять полученные знания при решении различных задач;
- представлений о роли информатики и ИКТ в современном обществе, понимание основ правовых аспектов использования компьютерных программ и работы в Интернете.

Требования к предметным результатам освоения базового курса информатики в области алгоритмизации и программирования должны отражать:

базовый уровень:

- владение навыками алгоритмического мышления и понимание необходимости формального описания алгоритмов;
- владение умением понимать программы, написанные на выбранном для изучения универсальном алгоритмическом языке высокого уровня; знанием основных конструкций программирования; умением анализировать алгоритмы с использованием таблиц;
- владение стандартными приемами написания на алгоритмическом языке программы для решения стандартной задачи с использованием основных конструкций программирования и отладки таких программ; использование готовых прикладных компьютерных программ по выбранной специализации;
- сформированность представлений о компьютерно-математических моделях и необходимости анализа соответствия

модели и моделируемого объекта (процесса); о способах хранения и простейшей обработке данных; понятия о базах данных и средствах доступа к ним, умений работать с ними;

профильный уровень:

- овладение понятием сложности алгоритма, знание основных алгоритмов обработки числовой и текстовой информации, алгоритмов поиска и сортировки;
- владение универсальным языком программирования высокого уровня (по выбору), представлениями о базовых типах данных и структурах данных; умением использовать основные управляющие конструкции;
- владение навыками и опытом разработки программ в выбранной среде программирования, включая тестирование и отладку программ; владение элементарными навыками формализации прикладной задачи и документирования программ. [27]

1.2.2. Авторские методики по изучению программирования в школьном курсе информатики

Изучение алгоритмизации и программирования имеет три целевых аспекта:

1. Развивающий аспект, под которым понимается развитие алгоритмического и логического мышлений обучаемых.

Основные умения и навыки, соответствующие алгоритмическому стилю мышления, которые формирует информатика:

- умения и навыки планирования структуры действий, необходимых для достижения цели при помощи фиксированного набора средств;
- структурирование сообщений;
- понимание и использование формальных способов кодирования решения задачи;
- технические навыки и умения взаимодействия с компьютером;

- проектирование и построение информационных и компьютерных моделей;
- инструментирование всех видов деятельности;
- умение производить структурный анализ задачи, разбивать большие задачи на малые, сводить нерешенные задачи к решенным.

2. Практический аспект. Важной целью изучения предмета «Информатика и ИКТ» является получение учениками опыта построения и исследования моделей реальных объектов на компьютере, в частности, с использованием систем программирования. Фундаментом любых компьютерных технологий являются компьютерные модели, которые отображают реальные объекты, явления, человеческую деятельность в компьютерные информационные структуры; именно в своей алгоритмической части информатика имеет дело с формальным описанием внешнего мира.

3. Программистский аспект. Профессия программист является достаточно распространенной и престижной, изучение программирования в курсе информатики позволяет обучаемому испытать свои способности к такого рода деятельности [22].

В современном образовании основными авторскими программами при изучении информатики являются УМК Л.Л. Босовой, А.Г. Гейна К.Ю. Полякова, И.Г. Семакина и Н.Д. Угриновича.

УМК Н.Д. Угриновича в рамках темы «Основы алгоритмизации и объектно-ориентированного программирования» предполагает изучение трех языков программирования: OpenOffice.org Basic в операционных системах Windows и Linux, объектно-ориентированный Visual Basic в операционной системе Windows и объектно-ориентированный Gambas в операционной системе Linux. Использование языка OpenOffice.org Basic согласуется с заданиями основного государственного экзамена (ОГЭ), а объектно-ориентированные Visual Basic и Gambas используют современную технологию программирования, к тому же алгоритмическое

программирование входит в технологию объектно-ориентированного программирования. [35]

И.Г. Семакин, после приведения классификаций современных парадигм программирования [Приложение 2] отмечает, что классической, универсальной и наиболее распространенной является процедурная парадигма. Наибольшее количество существующих языков программирования относится к этой линии. Поэтому чаще всего в образовательных учреждениях изучается процедурное программирование. А наиболее часто изучаемыми в школе языками программирования являются Паскаль и Бейсик [31]. Это подтверждается тем, что УМК Л.Л. Босовой, А.Г. Гейна и К.Ю. Полякова так же используют Pascal в качестве изучаемого языка программирования.

В разделе базового курса «Введение в программирование» необходимо продолжать ту же структурную линию, которая была заложена в алгоритмическом разделе. Поэтому при выборе языка программирования следует отдавать предпочтение языкам структурного программирования [31].

Изучения программирования И.Г. Семакин, как и другие авторы образовательных программ, начинает с «программирования» на алгоритмическом языке.

Алгоритмический язык (АЯ) – это текстовая форма описания алгоритма. Она ближе к языкам программирования, чем блок-схемы. Однако это еще не язык программирования. Поэтому строгого синтаксиса в Алгоритмическом языке нет. Для структурирования текста алгоритма на АЯ используются строчные отступы. При этом соблюдается следующий принцип: все конструкции одного уровня вложенности записываются на одном вертикальным уровне; вложенные конструкции смещаются относительно внешней вправо. [31]

И.Г. Семакин утверждает, что целесообразно для начального знакомства с языками программирования использовать Pascal, так как расхождение в языке Pascal и алгоритмическим языком минимальны: АЯ – русскоязычный,

Pascal – англоязычный; синтаксис Pascal определен строго и однозначно в отличие от сравнительно свободного синтаксиса АЯ, но при этом в основу обоих языков заложен принцип поддержки структурной методики программирования. [20]

Таким образом, основным изучаемым языком программирования в рамках общего образования является Pascal, так как:

1. Является языком структурного программирования.
2. Имеет большое сходство с алгоритмическим языком программирования, который учащиеся используют при первоначальном написании алгоритмов для исполнителей в более младших классах.
3. Был специально создан в качестве учебного языка программирования.

Однако на примере УМК Н.Д. Угриновича мы видим, что с методической точки зрения не только Паскаль не является обязательным для обучения программированию, но и языки структурного программирования в целом. Напротив, автор использует объектно-ориентированные языки, реализующие современные технологии программирования.

В демо-версиях итоговых аттестаций (ОГЭ и ЕГЭ) 2018 года задания первой части, подразумевающие знание языка программирования (задание 9, 10 ОГЭ; задание 8, 11, 19, 20, 21 ЕГЭ), предлагают на выбор пять вариантов: алгоритмический язык, Бейсик, Паскаль, Python, C++. Так же проверяется умение оптимизировать и менять алгоритм без ущерба для конечного результата (задание 6 ОГЭ). Во второй части ЕГЭ задания, требующие знание языка программирования не дают четких указаний о необходимости использовать один из приведенных в качестве примеров языков программирования. Напротив, в задании 25 ЕГЭ указано, что учащийся может использовать любой язык программирования, указав название и используемую версию [13].

Согласно спецификаторам ОГЭ 2018 года в ходе экзамена проверяется:

- Знание:
    - понятия алгоритма, его свойств, способов записи;
    - основных алгоритмических конструкции (ветвление и циклы).
- Умение применять свои знания в стандартной ситуации:
    - использовать стандартные алгоритмические конструкции для построения алгоритмов для формальных исполнителей;
    - оценивать результат работы известного программного обеспечения.
- Умение применять свои знания в новой ситуации:
    - разрабатывать алгоритмы для формального исполнителя или на языке программирования с использованием условных инструкций и циклов, а также логических связок при задании условий [31].

Согласно спецификаторам ЕГЭ 2018 года в ходе экзамена проверяется:

- Умение применять свои знания в стандартной ситуации:
    - использовать стандартные алгоритмические конструкции при программировании;
    - формально исполнять алгоритмы, записанные на естественных и алгоритмических языках, в том числе на языках программирования.
- Умение применять свои знания в новой ситуации:
    - анализировать текст программы с точки зрения соответствия записанного алгоритма поставленной задаче и изменят его в соответствии с заданием;
    - реализовывать сложный алгоритм с использованием современных систем программирования [32].

Таким образом, ОГЭ и ЕГЭ проверяют знание элементарных алгоритмических конструкций, умения исполнять элементарные алгоритмы,

составлять алгоритмические конструкции и изменять их [24] и знание *любого* языка программирования.

Так как нет ни законодательных, ни методических правил при выборе языка программирования при обучении программированию школьников, преподаватель может использовать любой из языков программирования, отвечающий его методическим целям и задачам.

1.2.3. Современные языки программирования и требования к выбору языка

Несмотря на единые требования образовательного стандарта, у каждого учителя свой набор требований к языку программирования. Это может быть удобная, дружелюбная среда разработки; простой, интуитивный синстаксис; кроссплатформенность и другие. Все это не относится к методике, но является важным фактором, который влияет на выбор языка.

Чтобы обосновать требования к языку программирования, необходимо, в первую очередь, определить цель изучения программирования.

К сожалению, наличие четких целей не дает нам единого ответа на вопрос, какой язык программирования лучше изучать со школьниками, так как многие из современных и не только языков программирования я легкостью удовлетворяют каждый из указанных целевых аспектов.

Проанализировав ФГОС, авторские методики и мнение учителей информатики, в ходе исследования были выдвинуты следующие требования:

- язык программирования, выбранный для обучения, должен быть максимально приближен к алгоритмическому языку по своей структуре, так как он является наиболее легким для усвоения принципов программирования учащимися (УМК И.Г. Семакина);
- выбранный язык программирования должен поддерживать структурную и объектно-ориентированную парадигму программирования (УМК Н.Д. Угриновича);
- код, составленный на выбранном языке программирования, должен быть легко читаем;

- язык должен быть строго типизированным, ибо смешение целых чисел, вещественных чисел и текстовых переменных приводит у начинающих программистов к неправильному представлению о методах хранения данных в памяти компьютера [9];
- язык программирования должен иметь возможности:
    - вычисления всех элементарных математических функций;
    - выполнения всех арифметических операций ("+", "-", "*", "/");
    - реализации:
        - логических операций сравнения: ">", ">=", "=", "<>","<=", "<";
        - логических операций: "И", "ИЛИ", "НЕ";
    - наличие аналога логической функции: ЕСЛИ:ТО:ИНАЧЕ:;
    - наличие оператора, позволяющего реализовать цикл или хотя бы оператор безусловного перехода;
- выбранный язык программирования должен предоставлять возможность организации массивов: в перечне возможных заданий ЕГЭ указаны задачи на обработку данных конечных числовых последовательностей (массивов, списков).

Для выявления наиболее популярных современных языков программирования просмотрим наиболее популярные рейтинги в данной сфере.

1. GitHub опубликовал рейтинг самых популярных языков программирования по числу pull-запросов. В нем с большим отрывом лидирует JavaScript с 2,3 миллиона запросов, тогда как у следующего за ним Python – 1 миллион. Также в пятерку входят Java, Ruby и PHP [6, Рис. 11].
2. Stack Overflow для составления рейтинга опросил 64 000 разработчиков. Здесь лидером тоже стал JavaScript – на нем программирует 61,9% опрошенных. Следом расположились SQL,

Java, C#, Python. Правда, 72,6% респондентов занимаются веб-разработкой, поэтому такие результаты вполне ожидаемы [1, Рис. 12].

3. Существуют рейтинги, которые принимают в расчет сразу несколько параметров для оценки популярности. TIOBE учитывает количество специалистов, обучающих курсов, независимых поставщиков и поисковых запросов. Пятерка лидеров здесь выглядит по-другому: Java, C, C++, C#, Python [7, Рис. 13].
4. У работодателей наиболее востребованные языки, по версии Coding Dojo, – это SQL, Java, Python, JavaScript и C++ [5, Рис. 14].

Изучив несколько рейтингов наиболее популярных языков программирования и объединив их результаты, можно сказать, что наиболее популярными языками программирования на 2017 год являются: Java, Phyton, JavaScript, C++, C#, PHP, SQL, Ruby, C. Наиболее популярными языками программирования в 2018 году считаются Swift, Go, PHP, C++, Python, JavaScript, Java, C#, Objective-C, Rust [2].

Так как выбирается язык программирования для первого знакомства обучающихся средней школы с основами программирования, исключаем из рассматриваемого нами списка языков следующие:

- Swift и Objective-C – языки для разработки нативных приложений для iOS или Mac OS;
- Go – язык для создания различных высокоэффективных программ, лучше всего подходит для разработки web-приложений;
- PHP – язык реализации web-приложений;
- Rust – экспериментальный язык программирования, поэтому стремительно изменяется: нет четкой документации и инструкций по изучению и использованию языка;
- JavaScript – язык для создание интерактивных сайтов Интернете, мобильных приложений, игр, десктопных приложений.

Между тем, необходимо добавить в наш список Pascal, как наиболее распространенный язык программирования, изучаемый в школах в данный момент.

Из современных языков программирования анализу на соотвествие требованиям, выдвинутым ранее, были подвергнуты Python, C++, C#, Java, Pascal.

В качестве характеристик для сравнения были выбраны:

- сходство структуры с алгоритмическим языком;
- поддержка структурной парадигмы программирования;
- поддержка объектно-ориентированной парадигмы программирования;
- читаемость и понятность кода;
- наличие типизации данных;
- возможность работы с массивами.

Возможность реализации базовых алгоритмических конструкций, логических операций, арифметических действий и функций в качестве критерия в анализе не учитывалась, так как все современные языки программирования отвечают данному требованию.

Изучив данные анализа в Таблица 1, можно сделать следующие выводы:

Python наиболее приближен к алгоритмическому языку программирования. Не имеет строгих синтаксических правил, является более современным языком программирования, чем вызывает больший интерес у учащихся.

C++ и C# языки программирования, рассчитанные на заинтересованных пользователей, программистов. Не подходят для обучения основам программирования в базовом курсе информатики, так как в классах данной категории достаточно большое количество обучающихся не интересующихся программированием. Повышенная сложность данных языков, вызванная их специализированностью, может ослабить интерес к программированию.

Таблица 1

Анализ современных языков программирования

Язык программирования	**Python**	**C++**	**C#**	**Java**	**Pascal**
Сходство структуры с алгоритмическим языком	+	-	-	-	+
Поддержка структурной парадигмы программирования	+	+	+	+	+
Поддержка объектно-ориентированной парадигмы программирования	+	+	+	+	В некоторых версиях языка. В частности, версия языка PascalABC.Net поддерживает объектно-ориентированную парадигму программирования.
Читаемость и понятность кода	+	-	-	+	+
Типизация данных	При создании переменной ее тип не указывается в явном виде, но позже может быть преобразован к какому-либо типу специальными функциями.	Типы данных указываются при создании переменных, но переменные могут быть преобразованы к другому типу.			При создании переменной необходимо в явном виде указать ее тип. Тип переменной нельзя изменять в ходе работы программы.
Возможность работы с массивами	Массивов как таковых не существует, однако есть такой тип данных, как списки. В определенной мере, списки подобны массивам в C. Единственной разницей является то, что элементы одного списка могут иметь разные типы данных [4].	+	+	+	+

Java более читаемый язык программирования, чем C++ и C#. Поддерживает основные принципы программирования, используемые в этих языках, облегчая последующие обучение C++ и C#. Для успешной работы необходимо подключать дополнительные библиотеки или импортировать в проект уже существующие для их последующего использования. Многие функции, доступные в других языках без особого усилия, необходимо инициализировать перед использованием, что не облегчает обучение программированию школьников в базовом курсе информатики.

Pascal наиболее распространенный язык обучения программированию в наши дни. Неудобен своей строгой типизацией и структурой: для многих учеников это является существенной проблемой, отвлекающей от понимания работы программы, ее структуры и принципов ее составления. [23]

Таким образом, учитывая особенности результатов и требований ФГОС, язык программирования Python является наиболее оптимальным решением. К тому же, мотивация к изучению «устаревших» языков программирования, по мнению обучающихся, намного ниже, чем мотивация к изучению современных языков программирования с их возможностями, которые они предоставляют, облегчая процесс создания программ и делая его увлекательнее. Поэтому заинтересованные ученики наиболее охотно будут изучать молодой и перспективный Python, нежели Pascal. Для незаинтересованных в обучении программированию, Python максимально облегчит работу массой встроенных функций и простотой синтаксиса.

1.3. Самостоятельная работа учащихся в обучении информатике

1.3.1. Авторские методики организации самостоятельной работы учащихся

Чтобы выявить наиболее оптимальное решение для организации самостоятельной работы при обучении программированию, проанализируем

различный УМК по информатике, рассмотрим каким образом авторы предлагают организовать самостоятельную работу учащихся.

В курсе информатики средней школы самостоятельная работа учащихся может быть организованная не только в области программирования. Однако учебные планы образовательных программ, соответствующих ФГОС не предусматривают самостоятельную работу учащихся во внеурочное время. Методические рекомендации касаются организации учебной деятельности в классе.

Несмотря на наличие метапредметных результатов по изучению курса «владение основами самоконтроля, самооценки, принятия решений и осуществления осознанного выбора в учебной и познавательной деятельности» [10] в методических пособиях по УМК Л.Л. Босовой отсутствуют не только рекомендации по организации самостоятельной деятельности учащихся, но и планирование данной деятельности.

В УМК К.Ю. Полякова указано, что умение самостоятельно планировать пути достижения цели, в том числе альтернативные, осознанно выбирать наиболее эффективные способы решения учебных и познавательных задач решается в ходе выполнения проектных заданий в учебниках для 7, 8 и 9 классов в рамках изучения следующих тем: 7 класс «Обработка графической информации», «Мультимедиа»; 8 класс «Кодирование информации», «Подготовка электронных документов»; 9 класс «Компьютерные сети». [26] При этом проектных заданий в части изучения алгоритмизации и программирования не предусмотрено.

И.Г. Семакин и М.С. Цветкова предлагают организовать самостоятельную работу учащихся в форме выполнения лабораторных работ. В задачнике-практикуме, входящем в состав УМК, помимо заданий для индивидуального выполнения в ряде разделов (прежде всего связанных с освоением информационных технологий), содержатся задания проектного характера (под заголовком «Творческие задачи и проекты»). В методическом пособии для учителя даются рекомендации об организации коллективной

работы над проектами [30]. Организация самостоятельной работы учащихся с использованием лабораторных работ не предполагает развития самоконтроля: проверку выполнения заданий осуществляет учитель.

Исходя из определения, используемого нами, важными элементами самостоятельной работы учащихся являются развитие самоконтроля и выполнение заданий во внеурочное время. Как видно из анализа существующих УМК, большинство видов самостоятельной работы рассчитаны на занятия в классе. К тому же даже при наличии рекомендаций по организации самостоятельной работы, контроль за выполнением и оценивание результатов осуществляются учителем, что не дает в полной мере развивать у учащегося самоконтроль.

1.3.2. Анализ способов организации самостоятельной работы учащихся при изучении алгоритмизации и программирования

Существует несколько основных видов деятельности учащихся, с помощью которых можно организовать самостоятельную работу в ходе изучения базового курса информатики: лабораторные практикумы и проектная деятельность.

Лабораторные практикумы как вид самостоятельной работы, безусловно, могут иметь место в образовательном процессе, но не заменять самостоятельную работу учащихся во внеурочное время. На начальных этапах изучения информатики, организация самостоятельной работы на уроке будет являться хорошим способом по подготовке к дальнейшему самостоятельному обучению учащихся, развитию их самостоятельности. Однако такая форма организации самостоятельной работы, как уже говорилось ранее, подразумевает постоянный контроль за работой учащихся и результатами их деятельности со стороны учителя. Во-первых, это отнимает время учителя: проверка каждой работы занимает время. Во-вторых, это не исключает консультаций, которые будут необходимы учащимся при выполнении заданий. В-третьих, у учащихся нет необходимости оценивать себя

самостоятельно, анализировать причины неудачи, пытаться самостоятельно найти решение проблемы: рядом всегда находится учитель.

Проектная деятельность как вид самостоятельной работы на уроках информатики не всегда осуществима. Одно дело разработать небольшую презентацию, пусть даже с самостоятельным поиском информации по использованию средств, совсем другое – написать программу в условиях девяти часов на изучение программирования в 8 и 9 классе. Очевидно, проектная деятельность хоть и осуществляется во внеурочное время, также в конечном счете оценивается учителем и применима не во всех областях информатики.

Для повышения эффективности самостоятельных работ и развития творческих способностей студентов профессиональных колледжей по информатике А.И. Туракулова предлагает внедрить инновационный элемент – эвристические задания в учебный процесс, т.е. применить эвристические задания к самостоятельным работам. [34]

Эвристическое задание – учебное задание, имеющее целью создание учеником личного образовательного продукта с использованием эвристических способов и форм деятельности. Эвристические задания, направленные на развитие эвристических качеств личности учащихся, играют ключевую роль в организации эвристического обучения. Цель эвристического обучения состоит в том, чтобы предоставить ученикам возможность творить знания, создавать образовательную продукцию по всем учебным предметам, научить их самостоятельно решать возникающие при этом проблемы. Первичным в учебном процессе выступает познание учеником реальной действительности; после получения соответствующих знаний и опыта происходит изучение достижений человечества в этой действительности. Деятельность, ведущая к созданию детьми образовательных продуктов, обнаруживает и развивает их индивидуальные способности [37].

Главный признак эвристического задания – его открытость, т.е. отсутствие заранее известного результата его выполнения. Поэтому другое

название эвристических заданий – открытые задания. Другой признак эвристического задания – опора на творческий потенциал ученика, обеспечение развития его творческих (эвристических) способностей. Ещё признак – сочетание универсальной предметной основы задания и уникального его рассмотрения учеником. Т.е. в задании предлагается рассмотреть общий для всех объект (предмет), используя индивидуальные (личностные) особенности ученика. В результате обеспечивается уникальность создаваемого образовательного продукта – результата выполнения эвристического задания. [34]

Одной из разновидностей данного типа заданий может выступать алгоритмическая задача, требующая реализации с использованием изучаемого языка программирования. Каждая задача может быть решена несколькими способами. Найти свой, понятный ему – главная задача ученика. Главная задача учителя – дать возможность учащемуся самостоятельно найти решение и реализовать его, применить имеющиеся знания, определить недостающие. Еще Л.Н. Толстой заметил: «Если ученик в школе не научился сам ничего творить, то и в жизни он всегда будет только подражать, копировать…». Для самостоятельного творческого овладения знаниями следует формировать способность к открытиям нового в известном, содействовать превращению этой способности в инструмент человеческой деятельности во всех сферах жизни, как писал Ян Амос Коменский: «Правильно обучать юношество — это не значит вбивать в головы собранную из авторов смесь слов, фраз, изречений, мнений, а это значит — раскрывать способность понимать вещи, чтобы именно из этой способности, точно из живого источника, потекли ручейки, подобно тому как из почек деревьев вырастают листья, плоды, а на следующий год из каждой почки вырастет целая новая ветка со своими листьями, цветами и плодами». [17]

1.3.3. Анализ средств по организации самостоятельной работы в рамках изучения алгоритмизации и программирования

По-нашему мнению, эвристическое задание в виде задачи на программирование – наиболее продуктивный способ организации самостоятельной деятельности учащихся.

Исходя из предложенного определения самостоятельной работы и выбранного способа организации, выбранное средство должно удовлетворять следующим требованиям:

- автоматизированная проверка решения и быстрая обратная связь для учащихся;
- просмотр статистики выполнения задания учащимися для учителя;
- ответом на задание должна являться программа.

Для реализации самостоятельной работы, удовлетворяющим данным требованиям необходимо использовать онлайн ресурсы.

Один из самых распространенных видов онлайн средств с поддержкой автоматизированной проверкой являются различные платформы по созданию тестовых материалов. Некоторые из этих ресурсов поддерживают возможность загрузки файлов (anketolog.ru, onlinetestpad.com), что позволяет нам использовать их для отправки решения на задачу в виде файла. Однако, проверка таких файлов осуществляется учителем вручную, как и ответы на задания со «свободным ответом», в котором учащиеся могут писать текст программы. Использование ресурсов по составлению тестов для организации самостоятельной работы эвристического типа нецелесообразно и не удовлетворяет нашим требованиям.

Существуют так же другие способы организации онлайн взаимодействия с учащимися, которые относятся к дистанционным технологиям. Различные платформы по организации курсов. Большинство из них так же требует проверки работ учащихся вручную, что создает некоторую задержку при проверке и уменьшает эффект воспитания самоконтроля.

Однако, существует специализированный ресурс для изучения программирования. Stepic – российская образовательная платформа и конструктор бесплатных открытых онлайн-курсов и уроков. Данная платформа позволяет любому зарегистрированному пользователю создавать интерактивные обучающие уроки и онлайн-курсы, используя видео, тексты и разнообразные задачи с автоматической проверкой и моментальной обратной связью.

Использование данного ресурса для организации самостоятельной работы учащихся при изучении программирования дает возможность учителю:

- сформировать задание и тестовые значения для него, на которых платформа будет самостоятельно организовывать проверку работ учащихся;
- просматривать статистику о прохождении учащимися курса;
- редактировать содержание курса по мере необходимости;
- выбирать язык программирования из тех, что поддерживает платформа, на котором учащиеся могут писать решения задач;
- размещать дополнительный материал и тестовые задания;
- подключать других учителей как разработчиками, так преподавателями для курса, что дает возможность разработать наиболее продуманный курс совместно с коллегами и использовать его не на одном классе.

Учащиеся при работе со Stepic имеют возможность:

- выполнять задания в любом порядке;
- пропускать задания, выбирая подходящий для себя уровень сложности;
- сдавать работу на любом из предложенных языков программирования;
- получать мгновенную обратную связь: проверка задания начинается сразу при отправке работы и буквально через

несколько секунд учащийся может видеть прошло ли его решение тесты или нет.

Таким образом, представленные в УМК способы организации самостоятельной работы не удовлетворяют нашему понимаю и видению данной деятельности, т.к. требуют постоянного участия и контроля учителя. Исходя из этого необходимо найти систему организации самостоятельного обучения такую, чтобы учащийся мог проверить правильность выполнения задания без участия учителя. При этом, в случае ошибки, возникнет необходимость разобраться в имеющемся материале, переосмыслить его. Одним из видов заданий, способствующих развитию самостоятельности учащихся, является эвристическое задание, которое в рамках изучения линии алгоритмизации и программирования может быть выражено в форме классической задачи по программированию. Ресурсом, который в дальнейшем будет использоваться нами, является российская специализированная под обучение программированию онлайн платформа Stepic, которая удовлетворяет всем выдвинутым нами требованиям.

В результате анализа ФГОС и авторских методик преподавания информатики нами подтверждена необходимость организации самостоятельной работы учащихся при изучении программирования. Одним из наиболее подходящих способов организации данного вида деятельности является использование эвристического задания. Данный тип задания можно отнести к различным рассмотренным нами классификациям:

- упражнения по решению задач;
- задания, ориентированные на применение практических знаний;
- творческие работы и логически-поисковые задачи.

При изучении программирования в базовом курсе информатики язык, используемый для обучения, выбирается на усмотрение учителя. Несмотря на то, что большинство УМК рассматривает в качестве изучаемого языка программирования Pascal, мы рекомендуем использовать один из современных языков программирования: Python.

В качестве средства для организации самостоятельной внеурочной работы учащихся мы рекомендуем использовать Stepic, т.к. этот ресурс удовлетворяет нашим требованиям и специализирован под деятельность подобного рода.

ГЛАВА 2. Разработка методических рекомендаций по организации самостоятельной работы учащихся в процессе обучения программированию

2.1. Самостоятельная работа учащихся при обучении программированию на современном языке

Для организации самостоятельной работы необходимо:

- определить цели самостоятельной работы и содержание, т.е. темы, по которым будут выдаваться задания;
- определить необходимое количество уровней сложности и критерии по оценке сложности задания;
- определить необходимое количество заданий и подобрать задачи;
- сформировать задания в выбранной системе;
- провести инструктаж с учащимися по работе с системой.

2.1.1. Планирование самостоятельной работы

В результате изучения содержательной линии «Алгоритмизация и программирование», согласно кодификаторам и спецификаторам ОГЭ учащиеся должны демонстрировать знание основных типов алгоритмических конструкций и умение:

- исполнить линейный алгоритм, записанный на алгоритмическом языке;
- исполнить простейший циклический алгоритм, записанный на алгоритмическом языке;
- исполнить циклический алгоритм обработки массива чисел, записанный на алгоритмическом языке;
- осуществить разбиение задачи на подзадачи, вспомогательный алгоритм;
- осуществить обработку следующих объектов: цепочка символов, числа, списки, деревья.

Кодификаторы и спецификаторы ЕГЭ добавляют к вышеперечисленным требованиям умение исполнять рекурсивный алгоритм.

При разработке методических рекомендаций по организации самостоятельной работы учащихся при изучении программирования с использованием Python опираемся на содержание программы разработанной нами на основе примерной программы по информатике в 7-9 классах, рекомендованной ФГОС (Приложение 1).

Примерное тематическое планирование изучение содержательной линии «алгоритмизация и программирование» представлено в Таблица 2. В 8 и 9 классе предусмотрено по 5 часов на самостоятельную работу учащихся, что соответствует требованиям ФГОС, представленном в процентном соотношении 60/40.

Таблица 2.

Планирование курса алгоритмизации и программирования в 8-9 классах

№	**Тема**	**Количество часов**	
		аудиторных	**самостоятельная работа**
8 класс			
1.	Основные сведения о языке программирования Python	1	
2.	Организация ввода, вывода данных. Преобразование типов данных. Программирование линейных алгоритмов. Арифметические операторы	1	1
3.	Программирование разветвляющихся алгоритмов. Условный оператор. Многообразие способов записи ветвлений	1	1
4.	Программирование циклов с заданным условием продолжения работы	1	1
5.	Программирование циклов с заданным числом повторений	1	
6.	Базовые алгоритмы. Алгоритмы накопления. Алгоритмы поиска максимума/минимума	1	2
7.	Решение задач	1	
9 класс			
1.	Решение задач на компьютере	1	
2.	Списки. Одномерные списки целых чисел описание, заполнение, вывод списка. Встроенные методы работы со списками. Вычисление суммы элементов списка	1	2
3.	Последовательный поиск в списке.	1	
4.	Сортировка элементов списка. Встроенные методы для работы со списками	1	
5.	Конструирование алгоритмов	1	
6.	Запись вспомогательных алгоритмов. Рекурсивные алгоритмы.	1	2
7.	Алгоритмы управления.	1	1

2.1.2. Подбор заданий для самостоятельной работы

Основным элементом при организации самостоятельной работы учащихся является задание. В нашем случае это задача по программированию, решение которой будет представлять код на изучаемом языке программирования.

Самостоятельная работа учащихся не должна ограничиваться повторением уже изученного материала и решением типовых заданий, она должна давать каждому учащемуся возможность развить свои знания в данной области, совершенствовать умея, искать творческий подход к решению нестандартных заданий. Исходя из этого определим необходимое количество уровней сложности. Использовать один уровень сложности нецелесообразно: ученики в классе имеют разный уровень подготовки и способностей, при развивающем воздействии подобранных заданий на слабых учениках, для сильных учеников данный уровень сложности будет слишком легким и наоборот: при подходящем для сильных учеников уровне сложности, слабые не будут справляться с заданиями, и их мотивация к дальнейшему изучению программирования будет стремительно ослабевать. При использовании двух уровней сложности происходит большая дифференциация учащихся. Мы уже имеем возможность подобрать задания для слабых учеников отдельно, а для сильных выбрать более сложные задания. Однако, мы советуем использовать три уровня сложности: первый уровень для учащихся, не успевающих на уроке; второй – для основной массы учащихся, в целом справляющихся с заданиями во время занятий; третий – задачи повышенной сложности для тех, кому недостаточно знаний, полученных на уроке. В дальнейшем мы будем использовать три уровня сложности заданий.

Определить количестве уровней сложности недостаточно. Необходимо так же разработать критерии, согласно которым мы будем относить задание к той или иной категории. Критерии выбираются исходя из целей, которые ставит учитель перед, результатов, которые ожидаются после выполнения учащимися самостоятельной работы. В нашем случае, мы поставили перед

собой целью самостоятельной работы решение учащимися задач, требующих от них ранее неиспользованного подхода к решению или поиска дополнительной информации. Исходя из этого, критерии разделения заданий по уровням можно представить в виде таблицы:

	Новый подход к решению	Поиск дополнительной информации
1 уровень	-	-
2 уровень	+	-
3 уровень	+	+

Несмотря на то, что задачи первого уровня сложности не требуют поиска информации или нового способа решения, для не успевающих учеников задачи данного уровня в соотношении со сложностью других уровней и учащихся, для которых они предназначены, будут относительно равноценны.

При организации самостоятельной работы так же следует учитывать мотивирующую силу отметки и четко определить для учащихся критерии, по которым будет выставляться отметка. Наиболее объективным в данном случае является определить соотношение набранных баллов и отметки, которая будет выставляться за данную работу. При определении количества баллов и оценки за них, необходимо учитывать количество заданий на каждом из уровней и их сложность (количество баллов за каждое задание). Представим наши результаты так же в таблице.

	Количество заданий	Баллы за одно верно выполненное задание	Общая сумма баллов
1 уровень	3	2	6
2 уровень	2	3	6
3 уровень	1	4	4

За задания различного уровня сложности дается разное количество баллов, что логично, однако нужно внимательно отнестись к системе оценивания. При выборе школы 1, 2 и 3 балла, за выполнение сложного задания учащийся получает в три раза больше баллов. Поэтому нами выбрана шкала 2, 3, 4. Она так же не является самой оптимальной, однако при нашем количестве заданий на каждом уровне является вполне приемлемой. Определяя, что для отметки «отлично» необходимо набрать 10 баллов, мы ставим ученика в ситуацию, когда ему недостаточно решить задачи только

третьего уровня или только второго. При этом у учащегося остается выбор того, какие именно задачи решить. При данном максимальном необходимом количестве баллов отметку «отлично» могут получить даже не самые способные учащиеся, просто решив большее количество заданий. Таким образом, при выбранной системе оценивания, мы мотивируем учащегося выполнять задания, имеющиеся наибольший вес, но при этом не ставим его в условия, когда для получения высокой отметки ему нужно сделать самое сложное из заданий, все еще сохраняя относительную трудность для каждого ученика.

Основываясь на задачах, решаемых на уроках в классе и предъявленных критериях, подбираем задачи, которые будут использоваться в дальнейшем в качестве заданий для самостоятельной работы. Данные задачи можно как составить, используя различные источники, так и придумать самостоятельно. Количество заданий каждого уровня сложности для каждого раздела рекомендуется сделать одинаковым. Для нашего курса мы отобрали следующие задачи:

Линейные программы

Уровень 1

1. Дано расстояние в сантиметрах. Найти число полных метров в нем.
2. Дано трехзначное число. Вывести сначала последнюю цифру (единицы). Затем первую цифру (сотни).
3. Известна стоимость монитора, системного блока, клавиатуры и мыши. Сколько будут стоить **N** компьютеров из этих элементов?

Уровень 2

1. Найдите среднее геометрическое двух чисел (квадратный корень произведения). [8]
2. Даны координаты трех точек **A, B, C**. Найти сумму всех отрезков (**AB, BC, AC**).

Уровень 3

1. Напишите программу, которая на вход будет получать два значения: строку и ключ, а на выходе возвращать строку, зашифрованную путем применения функции **XOR** (^) над символами строки с ключом. [15]

Условный оператор

Уровень 1

1. Пользователь вводит месяц. Вывести к какому времени года относится.
2. Определить является ли введенный год високосным.
3. Написать программу, принимающую 3 аргумента: первые 2 — числа, третий — операция, которая должна быть произведена над ними. Если третий аргумент "+", сложить их; если "-", то вычесть; "*" — умножить; ":" — разделить (первое на второе). В остальных случаях вернуть строку "Неизвестная операция".

Уровень 2

1. Дано целое число. Вывести количество учеников в письменном виде (двенадцать учеников).
2. В восточном календаре принят 60-летний цикл, состоящий из 12-летних подциклов, обозначаемых названиями цвета: синий, красный, желтый, белый и черный. В каждом подцикле года носят названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. По номеру года вывести его название, если 1984 был началом цикла – годом зеленой крысы.

Уровень 3

1. Напишите программу, которая по введенным коэффициентам будет решать квадратное уравнение. Не забудьте предупредить пользователя, если происходит деление на 0 или уравнение не имеет корней.

Циклы

Уровень 1

1. Проверить, является ли введенное число числом Фибоначи.

2. Колхозница привезла на рынок корзину яиц для продажи. Продавала она их по одной и той же цене. После продажи яиц колхозница пожелала проверить, верно ли она получала деньги. Но вот беда: она забыла, сколько яиц у нее было. Вспомнила она только, что когда перекладывала яйца по 2, то осталось одно яйцо; одно яйцо оставалось так же при перекладывании яиц по 3, по 4, по 5 и по 6. Когда же она перекладывала яйца по 7, то не оставалось ни одного. Помогите колхознице сообразить, сколько у нее было яиц. [21]
3. Вывести квадраты всех чисел от 1 до **n.**

Уровень 2

1. Пользователь делает вклад в размере a рублей сроком на **years** лет под 10% годовых (каждый год размер его вклада увеличивается на 10%. Эти деньги прибавляются к сумме вклада, и на них в следующем году тоже будут проценты). Написать программу, по аргументам **a** и **years** возвращающую сумму, которая будет на счету пользователя.
2. Дано целое число **p**. Вывести все его цифры по одной в строке начиная с последней (самой правой).

Уровень 3

1. Данные целые положительный числа **A** и **B**. найти наибольший общий делитель, используя алгоритм Евклида.

Базовые алгоритмы (накопления/поиска максимума(минимума))

Уровень 1

1. Найти сумму и произведение цифр, введенного натурального числа.
2. Дано натуральное число. Вывести на экран все простые числа до заданного включительно. [12]
3. Дано натуральное число n (которое также может быть равно нулю). Вычислить n!

   Примечание: n! (факториал числа n, читается «эн факториал») – произведение всех натуральных чисел до n включительно.

Уровень 2

1. Дано натуральное число **n**. Вывести на экран **n** первых простых чисел.
2. Дано натуральное число **n**. Проверить, представляет ли собой палиндром его десятичная запись.

Уровень 3

1. Дано натуральное число. Проверить, является ли оно совершенным.
   Примечание: совершенным числом называется натуральное число, равное сумме всех своих собственных делителей (то есть натуральных делителей, отличных от самого числа). Например, 6 – совершенное число, оно имеет три собственных делителя: 1, 2, 3, и их сумма равна 1+2+3 = 6. [12]

Списки

Уровень 1

1. На вход программа получает количество элементов списка и сами элементы по порядку. После этого еще одно число. Определить, содержит ли список данное число **x**. Если содержит, то вывести на экран число **7145**, если не содержит, то вывести на экран число **5741**. [3]
2. На вход программа получает количество элементов списка и сами элементы по порядку. Найти наибольший элемент списка и вывести его на экран.
3. На вход программа получает количество элементов списка и сами элементы по порядку. Поменять местами самый большой и самый маленький элементы списка.

Уровень 2

1. Напишите программу, на вход которой подаётся список чисел одной строкой. Программа должна для каждого элемента этого списка вывести сумму двух его соседей. Для элементов списка, являющихся крайними, одним из соседей считается элемент, находящий на противоположном конце этого списка. Например, если на вход подаётся список «1 3 5 6 10»,

то на выход ожидается список «13 6 9 15 7». Если на вход пришло только одно число, надо вывести его же.

Вывод должен содержать одну строку с числами нового списка, разделёнными пробелом. [3]

2. Из списка чисел удалить элементы, значения которых больше **35** и меньше **65**. При этом удаляемые числа сохранить в другом списке.

Уровень 3

1. Дан список-массив, заполненный случайным образом нулями и единицами. Найти самую длинную непрерывную последовательность единиц и определить индексы первого и последнего элементов в ней. [29]

Вспомогательные алгоритмы, рекурсия

Уровень 1

1. Напишите функцию, которая будет генерировать список заданного размера и заполнять его случайными числами (**make_list()**), а также функцию, которая будет выводить заданный список (**print_list()**).
2. Напишите программу, которая на вход будет получать количество координат и координаты точек (**x, y**) и выводить сумму всех отрезков.
3. Напишите функцию, которая определяет количество разрядов введенного целого числа.

Уровень 2

1. В теории вычислимости важную роль играет функция Аккермана **A(*m*,*n*),** определенная следующим образом:

$$A(m,n) = \begin{cases} n+1, & m = 0; \\ A(m-1,1), & m > 0, n = 0; \\ A\big(m-1, A(m,n-1)\big), & m > 0, n > 0. \end{cases}$$

Даны два целых неотрицательных числа **m** и **n**. Выведите **A(*m*,*n*)**. [28]

2. Пользователь вводит число. Сообщить, есть ли оно в массиве, элементы которого расположены по возрастанию значений, а также, если есть, в

каком месте находится. При решении задачи использовать бинарный (двоичный) поиск, который оформить в виде отдельной функции.

Уровень 3

1. Написать рекурсивную процедуру, переводящую числа из одной системы счисления в другую.

Таким образом, перед созданием ресурса для самостоятельной работы, необходимо проделать предварительную подготовку: определиться с целями самостоятельной работы, количеством и составом заданий, системой оценивания задач и самостоятельной работы.

2.2. Методические рекомендации по организации самостоятельной работы учащихся при обучении программированию с использованием онлайн платформы

2.2.1. Создание и редактирование курса и его содержания

Для организации самостоятельной работы в учащихся 8-9 классов в процессе обучения программированию мы рекомендуем использовать онлайн платформу Stepic. Данная платформа позволяет бесплатно разрабатывать и публиковать курсы, а также проходить их.

Для создания своего курса зарегистрируйтесь на сайте. В верхнем меню найдите «Создать» и выберете «Новый курс». Заполните первоначальные данные о курсе: введите его название и краткое описание (Рис. 1.)

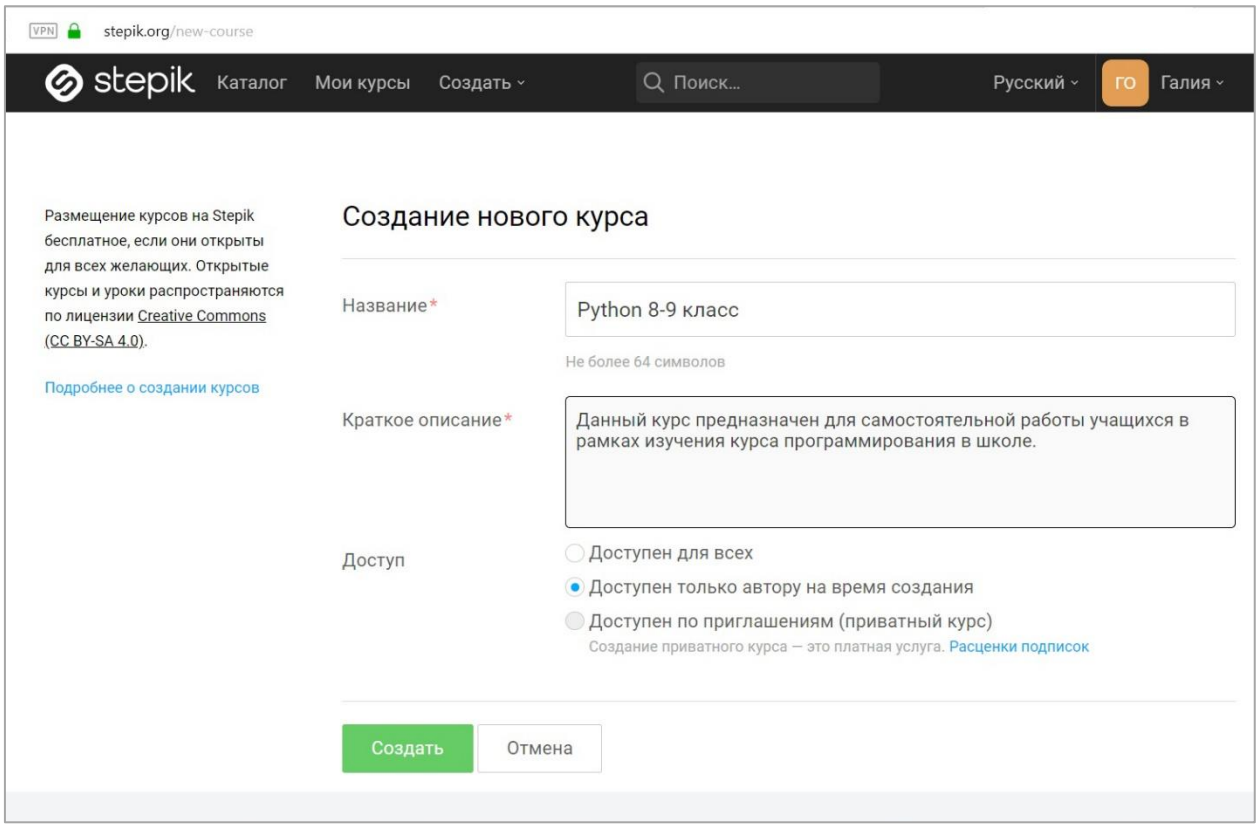

Рис. 1. Создание курса

Далее наполните свой курс содержанием. Создайте модули, по которым у вас предусмотрена самостоятельная работа (Рис. 2). В нашем случае это:

1. Линейные программы.
2. Условия.
3. Циклы.
4. Строки.
5. Списки.

Обратите так же внимание на то, что вы можете назначать время действия курсе, т.е. когда учащиеся смогут выполнять задания.

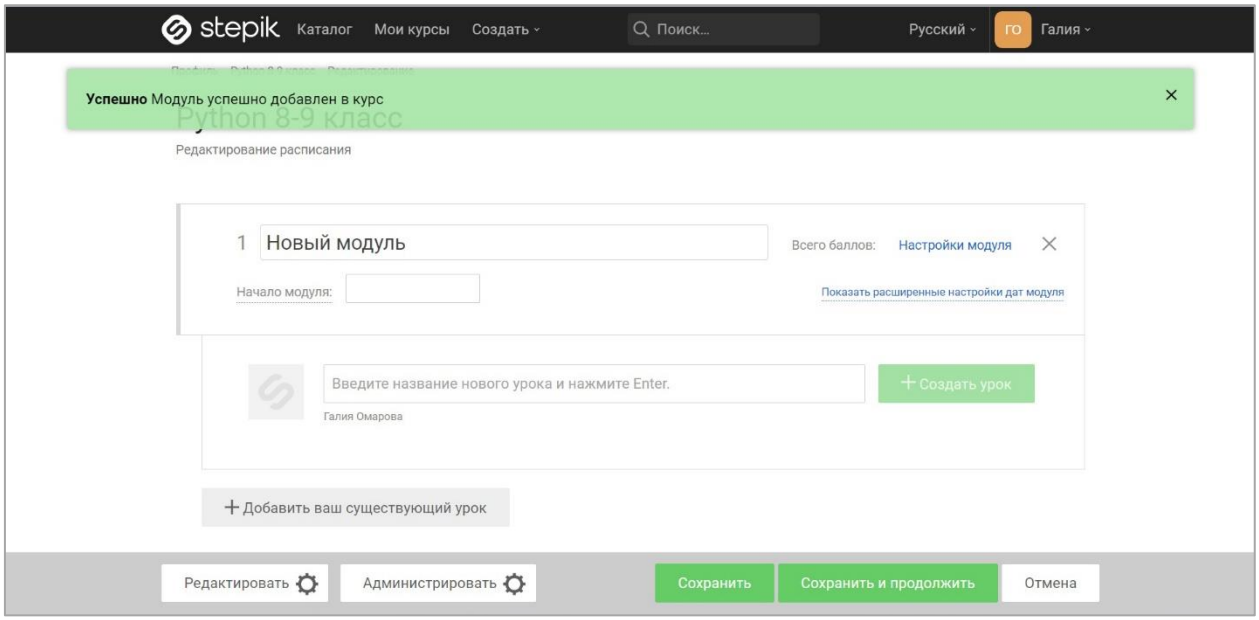

Рис. 2. Редактирование содержания курса

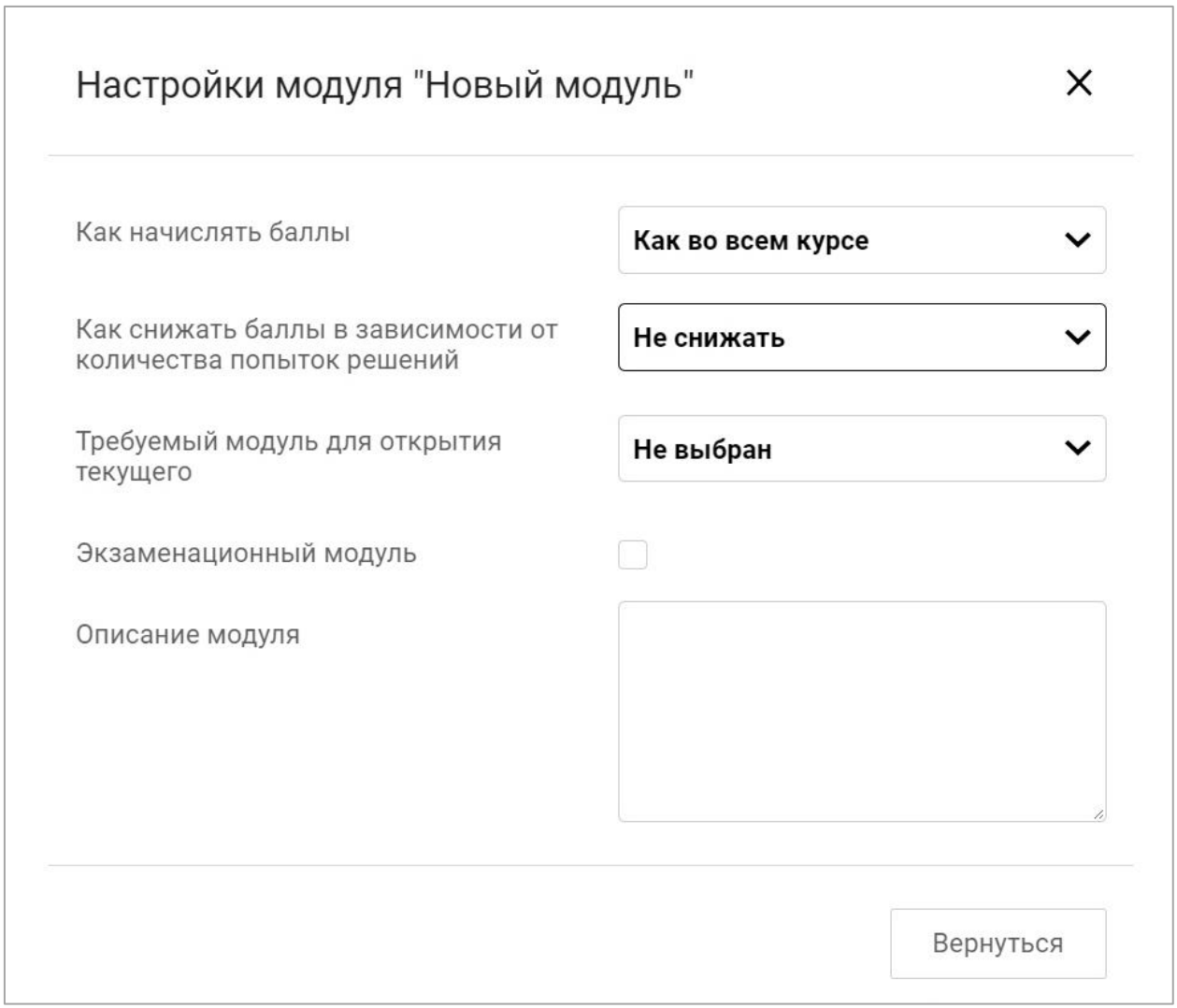

Рис. 3. Настройка модуля курса

Выберите настройки для каждого модуля (Рис. 3). Определите, будут ли снижаться баллы за количество попыток, использованных для получения правильного решения. Нужно ли выполнить какой-то другой модуль, чтобы получить доступ к выбранному. При необходимости укажите дату начала модуля (когда задания будут доступны для работы с ними). Обратите снимание, что дата модуля имеет скрытые настройки. В них вы можете указывать даты, когда подразумеваете окончание работы с данным модулем, и дату окончания модуля.

Далее переходим к созданию уроков. Каждый урок может состоять и нескольких объектов. На данном этапе нам нужно всего лишь ввести название. Для того, чтобы учащимся было легче ориентироваться, используем в качестве названия определенные нами уровни сложности. После нажатия на кнопку «Создать урок», ярок будет добавлен в список, а вам предоставлена возможность создавать другие уроки. Не забывайте сохранять ваши результаты. Рекомендуем использовать вариант «Сохранить и продолжить». При использовании варианта «Сохранить» вы автоматически выйдете из редактора. Если это произошло, вернуться можно с помощью клавиши «Редактировать содержание».

После того, как вы закончили с редактированием содержания, переходите к редактированию уроков. Каждый урок может состоять из нескольких шагов. Каждый шаг может иметь свой тип. Типы шагов представлены на панели слева. Создавать новый шаг можно как нажатием на нужный тип на этой панели, так и выбором типа шага после нажатия на кнопку «Добавить новый шаг», которая обозначена плюсом в конце линии шагов.

При первом открытии редактора урока автоматически создается шаг типа текст. При необходимости его, как и любой другой шаг, можно удалить. Мы рекомендуем использовать первым шагом тип «текст» для размещения новой информации, необходимой при решении задач, или ссылок на источники, в которых эту информацию можно найти. Так же вы можете использовать различные типы шагов для проведения тестирования или

организации контрольных мероприятий, однако мы подробнее остановимся на шаге типа «Программирование», т.к. он является основным видом задания при обучении программированию.

2.2.2. Создание задания по программированию

Вкладка редактирования задания по программированию состоит из двух блоков: условия и настройки.

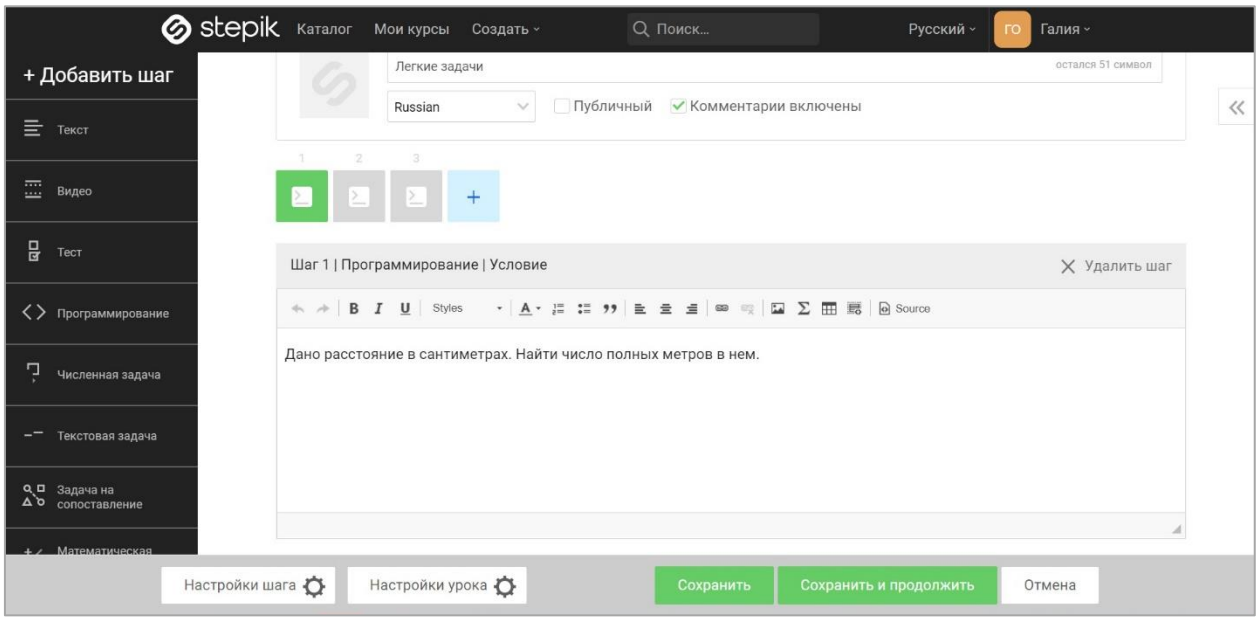

Рис. 4. Создание и редактирование условия задачи

Разместите в блоке «Условие» задачу. Отредактируйте с помощью встроенного редактора при необходимости (Рис. 4).

Перейдите к блоку «Настройки». Данный блок состоит из вкладок «тестовые данные», «языки и шаблоны», «расширенный редактор».

В верхней части блока выберите максимальное количество баллов, которое может получить учащийся за правильно решенную задачу.

В онлайн платформе Stepic существует несколько способов проверки задачи. Первый – использование известных тестовых значений, заданных вручную (Рис. 5). Мы рекомендуем использовать данный тип тестирования на несложных задачах, не требующих длительных вычислений и большого количества входных и выходных данных. В поле «input» введите значения, которые программа будет получать на вход. Рекомендуется оставлять пустую строку после перечисления значений. В поле «output» введите значение,

которое должна выдавать корректно работающая программа при введенных ранее входных значениях. Каждый тест создается и заполняется отдельно.

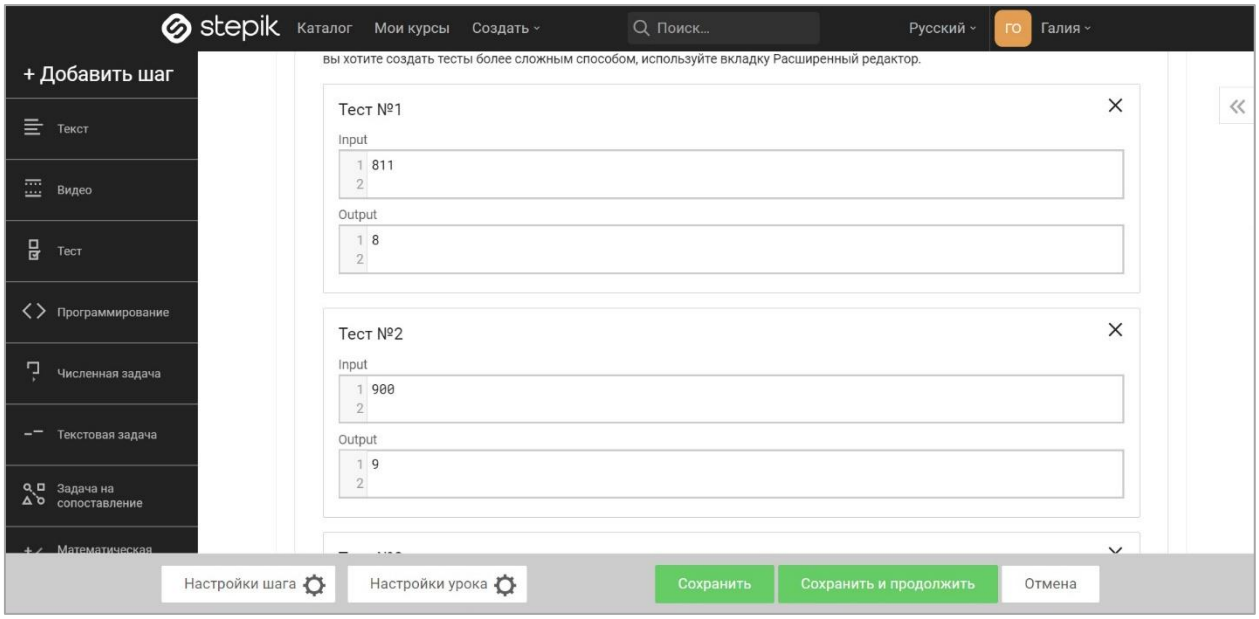

Рис. 5. Создание тестов. Способ 1

Данный способ не удобен, если вам требуется проверить задачу на большом наборе значений. В этом случае воспользуйтесь вкладкой «Расширенный редактор». Данный редактор является исполняемым кодом, состоящим из трех функций, написанных на языке Python. Функция **generate()** генерирует список входных данных. Функция **solve()** получает на вход сгенерированные данные и высчитывает выходные. Функция **check()** проверяет корректность работы сданной программы. Обращаем ваше внимание, что для корректной работы данного метода проверки заданий вкладка «Тестовые задания» должна быть пустой. Представим пример проверки третьей задачи первого уровня сложности и модуля «Линейные программы».

```
def generate():
    import random
    num_tests = 10
    tests = []
    for test in range(num_tests):
        random.seed
        monitor = random.randint(2000, 4000)
```

```
            block = random.randint(20000, 50000)
            klaviatura = random.randint(1000, 5000)
            mouse = random.randint(500, 2000)
            comps = random.randint(0, 100)
            test_case    =    "{}\n{}\n{}\n{}\n{}\n".format(monitor,
block, klaviatura, mouse, comps)
            tests.append(test_case)
        return tests

    def solve(dataset):
        m, b, k, ms, c = dataset.split()
        p = int(m) + int(b) + int(k) + int(ms)
        p = p * int (c)
        return str(p)

    def check(reply, clue):
        reply, clue = int(reply), int(clue)
        if reply == clue:
            return True
        feedback = "You answer was: {}. Correct answer was:
{}".format(reply, clue)
        return False, feedback
```

При любом типе тестирования задач, максимальное количество тестов 100.

Перейдем к вкладке «Языки и шаблоны». В данной вкладке можно ограничить набор языков программирования, которые учащиеся могут использовать при выполнении задания (Рис. 6). Онлайн платформа Stepic поддерживает проверку кода на следующих языках программирования:

- ASM32;
- ASM64;
- C;
- C#;
- C++, C++11;

- Clojure;
- Go;
- Haskell (ghc 7.8), Haskell (ghc 7.10), Haskell (ghc 8.0);
- Java 7, Java 8, Java 11;
- JavaScript;
- Octave;
- PascalABC.NET;
- Perl;
- Python 3;
- R;
- Ruby;
- Rust;
- Scala;
- Shell.

Если вы хотите ограничить языки, на которых будет приниматься решение конкретной задачи, в данной вкладке укажите название языка маленькими буквами после двух знаков двоеточия.

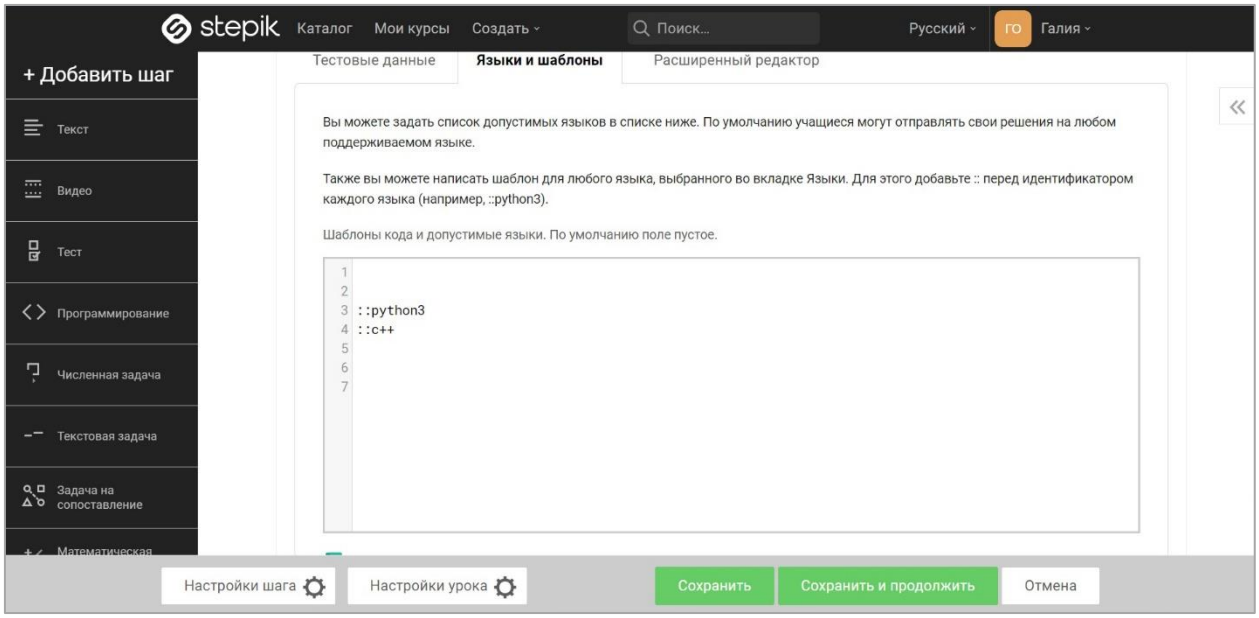

Рис. 6. Задание языков программирования при проверке

Кроме того, можно указать шаблон кода решения для каждого языка. Шаблон выводится в поле для решения при выборе языка. Шаблон кода может

быть полезен для определения начального кода программы, который необходимо дополнить учащемуся до правильного решения. Например, могут быть определены вспомогательные функции, написан ввод/вывод данных, задана схема класса или функции, которую необходимо реализовать учащемуся. В чистом виде шаблон кода не обязан быть корректной программой.

Когда с набор тестовых заданий готов и необходимые шаблоны заданы, перейдите к полному писку настроек, который находится под вторым блоком.

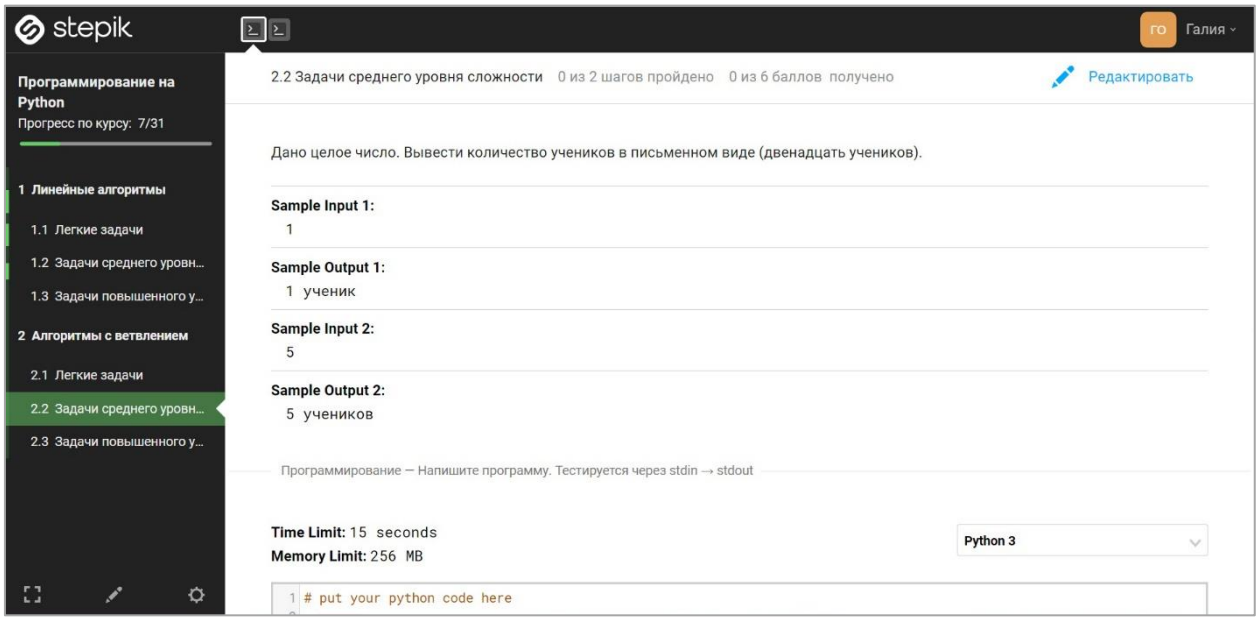

Рис. 7. Пример обзора тестовых параметров учащимся

В данном списке при необходимости укажите лимит времени на исполнение кода программы учащегося, лимит памяти, задействованный под хранение кода данных в ходе выполнения программы (не учитывает размер кода программы), а также количество тестов, которые будут видны учащемуся. По умолчанию учащихся видит один тест. При необходимости, измените данное значение (Рис. 7). Например, в задаче с определением является ли год високосным, логично показать учащемуся два теста с различным выводом. В этом же разделе ограничьте количество попыток на сдачу задания при необходимости.

В случае, если вы составляете курс не один, откройте доступ коллегам. Откройте главную страницу курса, откройте меню рядом с кнопкой

«Продолжить» и выберите «Доступ к курсу». Скопируйте ссылку-приглашение и отправьте коллегам или добавьте их по id.

Исходя из вышеописанного, видно, что публикация заданий не требует особых затрат от учителя. Необязательно придумывать набор тестов, достаточно самому решить задачу и сгенерировать входные данные.

2.3. Апробация

Апробация разработанного материала проводилась методом экспертных оценок. Экспертами выступали студенты Уральского государственного педагогического университета Института математики, физики информатики и технологий, в количестве 7 человек. Экспертам были представлены методические рекомендации. Целью анкет было оценить понятность и подробность разработанных рекомендаций.

Экспертам были заданы следующие вопросы:

1. Возникли ли трудности с формулировками?
   - ☐ Да
   - ☐ Нет
2. Ответы на все возникшие вопросы присутствуют в методических рекомендациях?
   - ☐ Да
   - ☐ Нет
3. Сможете ли вы, используя данные рекомендации, составить свою программу самостоятельной работы?
   - ☐ Да
   - ☐ Нет
4. Есть ли какие-либо пожелания с целью улучшения методических рекомендаций?

Результаты анкетирования представлены в диаграммах:

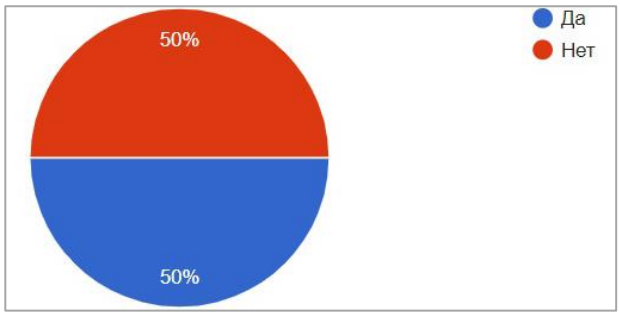

Рис. 8. Вопрос 1

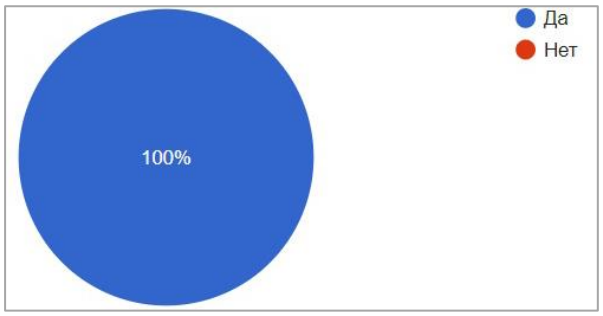

Рис. 9. Вопрос 2

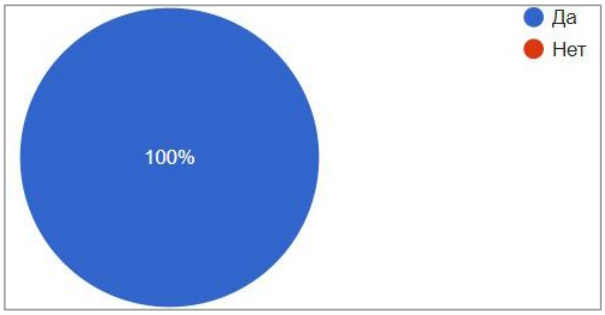

Рис. 10. Вопрос 3

В результате анкетирования были проведены работы по корректированию методических рекомендаций.

Таким образом были разработаны методические рекомендации по составлению курса с использованием онлайн платформы Stepic и рассмотрены общие рекомендации разработки курсов по самостоятельной работе учащихся в рамках учения линии «Алгоритмизации и программирования».

Заключение

В результате выполнения работы были проанализированы нормативные документы основного общего образования (ФГОС) на наличие требований к изучаемым языкам программирования и авторские учебные программы по информатике и ИКТ. Так же были изучены кодификаторы и спецификаторы основного и единого государственных экзаменов. В результате исследования было обнаружено, что на законодательном уровне выбор изучаемого в школе языка программирования не регламентируется. Однако существуют параметры, которым изучаемый язык должен соответствовать для достижения образовательных целей, установленных федеральным стандартом.

Был проведен сравнительный анализ современных распространенных языков программирования на степень возможности обучения на них (соответствие выдвинутым критериям). В результате анализа наиболее оптимальным вариантом для обучения программированию детей оказался язык программирования Python.

В результате рассмотрения видов и способов организации самостоятельной работы учащихся, а также анализа нормативных документов на наличие требований к организации самостоятельной работы основным способом организации самостоятельной работы при изучении программирования был предложен метод эвристического задания. После анализа средств для организации самостоятельной работы учащихся было предложено организовать самостоятельную работу учащихся с использованием онлайн-платформы Stepik.

Цель работы была достигнута. Были разработаны и апробированы методические рекомендации по организации самостоятельной работы учащихся в средней школе при обучении программированию с использованием современных языков программирования. Так же по результатам работы было опубликовано две статьи.

Данная разработка может быть полезна как начинающим учителям информатики, так и опытным учителям, решившим начать преподавание нового языка программирования.

В результате выполнения все задачи были выполнены, цели достигнуты.

Библиографический список

1. Developer Survey Results // Stack Overflow URL: https://insights.stackoverflow.com/survey/2017 (дата обращения: 28.05.2019).
2. proglib // 10 лучших языков программирования для изучения в 2018 году URL: https://proglib.io/p/10-languages-2018/ (дата обращения: 28.05.2019).
3. PYTHON. Задачи на списки // Информатика Эксперт URL: http://informatikaexpert.ru/python-zadachi-na-spiski/ (дата обращения: 28.05.2019).
4. Pythonic way URL: http://pythonicway.com/ (дата обращения: 28.05.2019).
5. The 9 Most In-Demand Programming Languages of 2017 // Coding DOJO blog URL: https://www.codingdojo.com/blog/9-most-in-demand-programming-languages-of-2017 (дата обращения: 28.05.2019).
6. The fifteen most popular languages on GitHub // GitHub URL: https://octoverse.github.com (дата обращения: 28.05.2019).
7. TIOBE Index for May 2018 // TIOBE URL: https://www.tiobe.com/tiobe-index// (дата обращения: 28.05.2019).
8. Абрамян М.Э. 1000 задач по программированию. Часть I Скалярные типы данных, управляющие операторы, процедуры и функции. Ростов-на-Дону: 2004.
9. Бобров А.Н. Проблемы выбора языка программирования в школьном курсе информатики // Молодой ученый. 2015. №24
10. Босова Л.Л., Босова А.Ю. Информатика. 7 – 9 классы. Методическое пособие. М.: БИНОМ. Лаборатория знаний, 2016.
11. Буряк В.К. Самостоятельная работа учащихся. М.: Просвещение, 1984.-64с.
12. Д. Душистов "Решение 50 типовых задач по программированию на языке Pascal" // URL: http://msk.edu.ua/ivk/Informatika/Books/Programmirovanie/Dushistov_Reshenie_50_tip_zadach_na_Pascal/Dushistov_Reshenie_50_tip_zadach_na_Pascal.pdf (дата обращения: 28.05.2019).

13. Демонстрационный вариант КИМ ЕГЭ // ФЕДЕРАЛЬНЫЙ ИНСТИТУТ ПЕДАГОГИЧЕСКИХ ИЗМЕРЕНИЙ URL: https://drive.google.com/file/d/0B8MkXVdvfYcrZWFVSmZhbW9DR1U/view (дата обращения: 28.05.2019).
14. Задачи на списки для освоения языка Python // Nikulux URL: http://nikulux.ru/python-praktika/zadachi-na-spiski-dlya-osvoeniya-yazyka-python/ (дата обращения: 28.05.2019).
15. Задачи по Python // Python 3 для начинающих URL: https://pythonworld.ru/osnovy/tasks.html (дата обращения: 28.05.2019).
16. Зимняя И.А. Педагогическая психология: Учеб. Для вузов. М.: Логос, 2004.
17. Кларин В.М., Джуринский А.Н. Педагогическое наследие. М.: Педагогика, 1989.
18. Коноводова Ю.А. Отличие самостоятельной деятельности учащихся от самостоятельной работы учащихся // Проблемы и перспективы развития образования: материалы Междунар. науч. конф. Пермь: Меркурий, 2011.
19. Коротяев Б.И., Пидкасистый П.И. Организация деятельности ученика на уроке. 6 изд. М.: Знание, 1985.
20. Лапчик М.П., Семакин И.Г., Хеннер Е.К. Методика преподавания информатики: учеб. пособие для студ. пед. вузов. М.: издательский центр «Академия», 2007
21. Нагибин Ф.Ф., Канин Е.С. Математическая шкатулка. 5 изд. М.: Просвещение, 1988.
22. Николаева И.В., Давлетярова Е.П. Теория и методика обучения инфрматике. Владимир: Изд-во ВлГУ, 2012.
23. Омарова Г.Р., Шимов И.В. Современные языки программирования при обучении программированию школьников // Актуальные вопросы преподавания математики, информатики и информационных технологий. Екатеринбург: Уральский государственный педагогический университет, 2018.

24. Особенности ОГЭ по информатике 2017 // Uchebu.com URL: https://prouchebu.com/oge_informatika_2017/ (дата обращения: 28.05.2019).
25. Пидкасистый П.И. Самостоятельная деятельность учащихся. М.: Педагогика, 1972.
26. Поляков К.Ю., Еремин Е.А. ИНФОРМАТИКА 7–9 классы. Методическое пособие. М.: БИНОМ. Лаборатория знаний, 2016
27. Приказ "Об утверждении федерального государственного образовательного стандарта среднего общего образования (с изменениями на 29 июня 2017 года)" от 17 мая 2012 № 413 // Собрание законодательства Российской Федерации.
28. Рекурсия. Тренировочные задачи // Habr URL: https://habr.com/ru/post/275813/ (дата обращения: 28.05.2019).
29. Решение задач на Python // Лаборатория линуксоида URL: https://younglinux.info/python/task (дата обращения: 28.05.2019).
30. Семакин И.Г., Цветкова М.С. ИНФОРМАТИКА. 7–9 классы. Примерная рабочая программа. М.: БИНОМ. Лаборатория знаний, 2016.
31. Семакин И.Г., Шейн Т.Ю. Преподавание базового курса информатики в средней школе. М.: БИНОМ. Лаборатория знаний, 2007.
32. Спецификатор ЕГЭ "Демоверсии, спецификации, кодификаторы ЕГЭ 2018 г." Федеральный институт педагогических измерений. 2018 г.
33. Спецификатор ОГЭ "Демоверсии, спецификации, кодификаторы ЕГЭ 2018 г." Федеральный институт педагогических измерений. 2018 г.
34. Туракулова А. И. Повышение эффективности самостоятельных работ по информатике посредством технологий эвристического обучения // Молодой ученый. Казань: 2012 №5.
35. Угринович Н.Д., Самылкина Н.Н. ИНФОРМАТИКА 7–9 классы. Примерная рабочая программа. М.: БИНОМ. Лаборатория знаний, 2016
36. Хуторской А.В. Дидактическая эвристика. Теория и технология креативного обучения. М.: Изд-во МГУ, 2003.

37. Цукерман Г.А., Венгер А.Л. Развитие учебной самостоятельности средствами школьного образования // Московский городской психолого-педагогический университет, 2010.

Приложения

Приложение 1

Таблица 3

Тематическое планирование и методические рекомендации по преподаванию программирования на языке программирования Python для 8-9 классов

Номер урока	Тема урока	Примерное содержание урока	Рекомендации/особенности, на которые стоит обратить внимание
8 класс. Начала программирования			
1.	Основные сведения о языке программирования Python	– История создания – Сферы использования – Алфавит и словарь языка – Используемые типы данных и их ограничения в Python – Структура программы – Переменные и их тип – Оператор присваивания	– Структура программы задается с помощью отступов, показывающих уровень вложенности операций – Тип переменной не указывается в явном виде – Поддерживается работа с числами в нескольких системах счисления – Существует оператор множественного присваивания. Используется, когда нескольким переменным необходимо присвоить одно и то же значение
2.	Организация ввода, вывода данных. Преобразование типов данных. Программирование линейных	– Вывод данных оператор print – Вывод строки – Вывод переменной – Вывод «склеенной строки» – Перевод строки при выводе	– Перевод строки осуществляется автоматически после выполнения каждого оператора print – При задании текста могут быть использованы как одинарные, так и двойные кавычки. Главное, чтобы они были парными

	алгоритмов. Арифметические операторы	– Ввод данных с помощью оператора input – Преобразование типов переменных, с чем это связано – Первая программа – Форма записи арифметических операторов – Особенности языка – Составление линейных программ	– Оператор input возвращает строку, поэтому для получения числа, необходимо преобразовать строковый тип данных к другому – Оператор input так же может выводить текст подсказки. Нет необходимости сначала выводить подсказку, затем организовывать ввод данных – Одновременное выполнение арифметической операции и присваивания
3.	Программирование разветвляющихся алгоритмов. Условный оператор. Многообразие способов записи ветвлений	– Оператор if – Синтаксис – Примеры решения задач выбора из двух альтернатив – Решение различных задач с использованием оператора if для выбора из трех и более вариантов	– С помощью данного оператора можно осуществлять выбор не только между двумя объектами, но и большим количеством – Обратить особое внимание на использование переменных различных типов данных. При сравнении двух переменных они не могут иметь тип double из-за особенности представления вещественных чисел в памяти компьютера. Так же специфика сравнение строковых переменных. – Трехместное выражение if/else (A = Y if X else Z)
4.	Программирование циклов с заданным условием продолжения работы	– Оператор while – Синтаксис – Примеры решения задач	– Оператор continue – Оператор break – Оператор else

5.	Программирование циклов с заданным числом повторений	– Оператор for – Синтаксис – Функция range () для генерации арифметической последовательности – Примеры решения задач	– В качестве итерируемого объекта может использоваться строка или список – Три варианта задания функции range и принцип ее работы – При разборе функции range понятие списка не дается – Обратить особое внимание на использование переменных различных типов данных при формировании логического выражения
6.	Базовые алгоритмы. Алгоритмы накопления. Алгоритмы поиска максимума/минимума	– Технология решения задач на накопление суммы, произведения, количества. – Технология решения задач на нахождение минимального/максимального значения. – Решение одной задачи с использованием обоих вариантов циклов	– Имитация работы цикла с заданным числом повторений и помощью оператора while – Решение задач на накопление суммы, произведения и количества – Нахождение максимального и минимального числа из трех
7.	Решение задач	– Решение комбинированных задач на использование нескольких операторов одновременно	– Оптимизация алгоритма
9 класс. Алгоритмизация и программирование			
1.	Решение задач на компьютере	– Повторение курса по программированию за 8 класс путем решения простых задач с использованием основных	

		алгоритмических конструкций и типов данных	
2.	Списки. Одномерные списки целых чисел описание, заполнение, вывод списка. Встроенные методы работы со списками. Вычисление суммы элементов списка	– Понятие списка – Способы создания списка – Способы работы с элементами списка – «Методы» списков – Решение задач на вычисление суммы элементов списка, содержащего числа	– Создание списков, состоящих из элементов одного типа и разных типов – Обращение объектов в списки (например, строки) с использованием стандартных функций – Генерация списков – Вычисление суммы строковых элементов списка – Использование различных циклов для решение задачи
3.	Последовательный поиск в списке.	– Решение задач на нахождение необходимого элемента в списке с использованием циклов и условных операторов	– Поиск наибольшего и наименьшего элемента списка – Использование стандартных методов при решении задач
4.	Сортировка элементов списка. Встроенные методы для работы со списками	– Алгоритм использования циклических конструкций для сортировки элементов списка – Использование встроенных методов для сортировки элементов списка	– Сортировка списка стандартным методом на основе различных функций – Сортировка элементов списка различных типов – Сортировка списка, состоящего из разнотипных элементов
5.	Конструирование алгоритмов	– Последовательное построение алгоритма – Понятие вспомогательного алгоритма	– Целесообразно на первом занятии для знакомства со вспомогательными алгоритмами использовать формального исполнителя

		– Понятие формальных и фактических параметров (аргументы функции) – Понятие рекурсивного алгоритма – Области применения	
6.	Запись вспомогательных алгоритмов. Рекурсивные алгоритмы.	– Технология записи вспомогательного алгоритма на Python – Решение задач	– Возвращение значение функции. Что может быть возвращено, в результате работы вспомогательного алгоритма. Возврат функции – Необаятельные аргументы функции – Переменное количество аргументов функции
7.	Алгоритмы управления.	– Понятие алгоритма управление – Обобщение и систематизация изученного материала	– Реализация алгоритмов управления при решении задач на циклические алгоритмические конструкции (вывод значения в зависимости от введенного параметра)

Приложение 2. Классификация парадигм программирования

К основным парадигмам программирования относятся:

- процедурное программирование (Паскаль, Бейсик, Фортран, C, ассемблеры);
- логическое программирование (Пролог);
- функциональное программирование (Лисп);
- объектно-ориентированное программирование (Smalltalk, C++, Delphy).

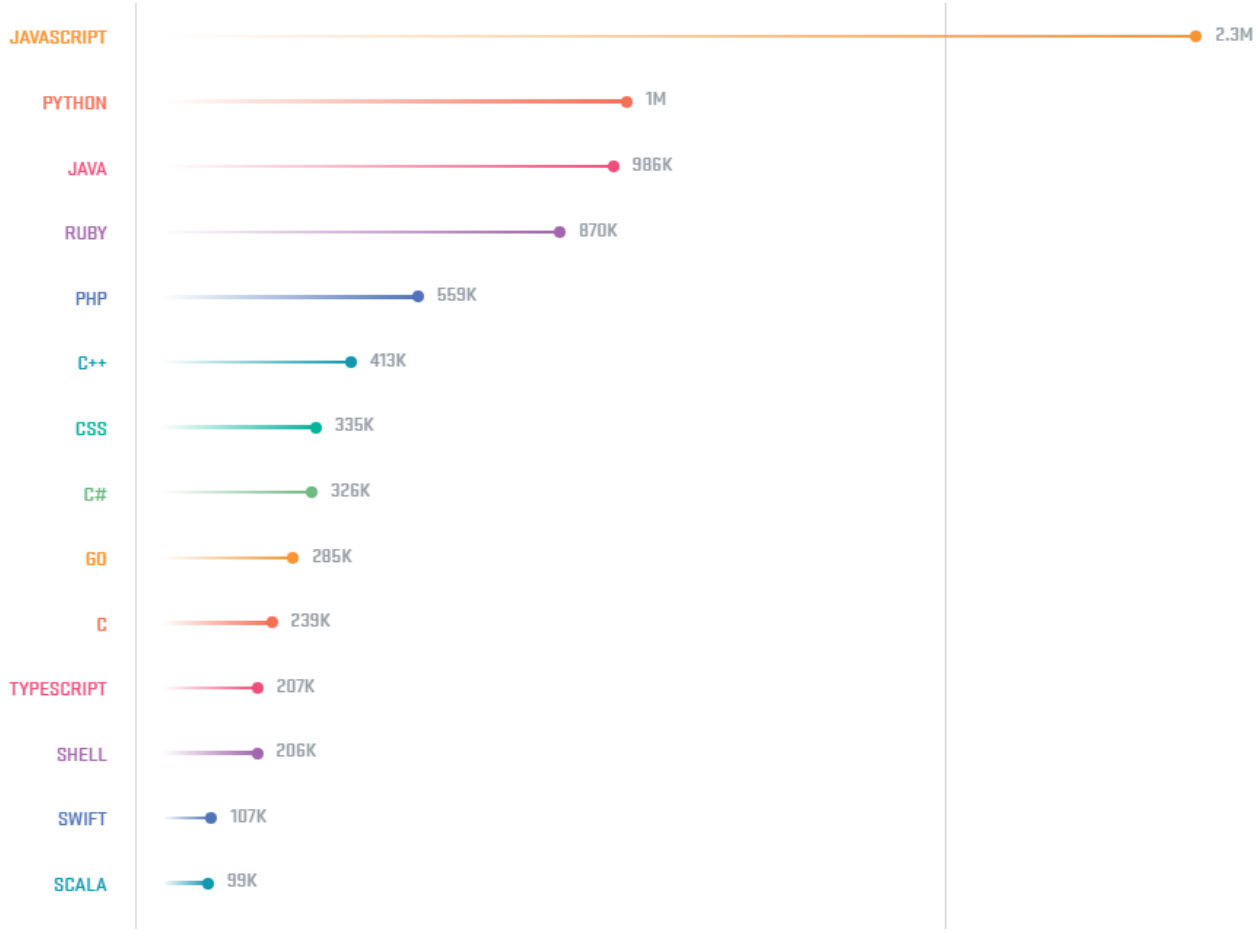

Рис. 11. Рейтинг самых популярных языков программирования по числу pull-запросов от GitHub.

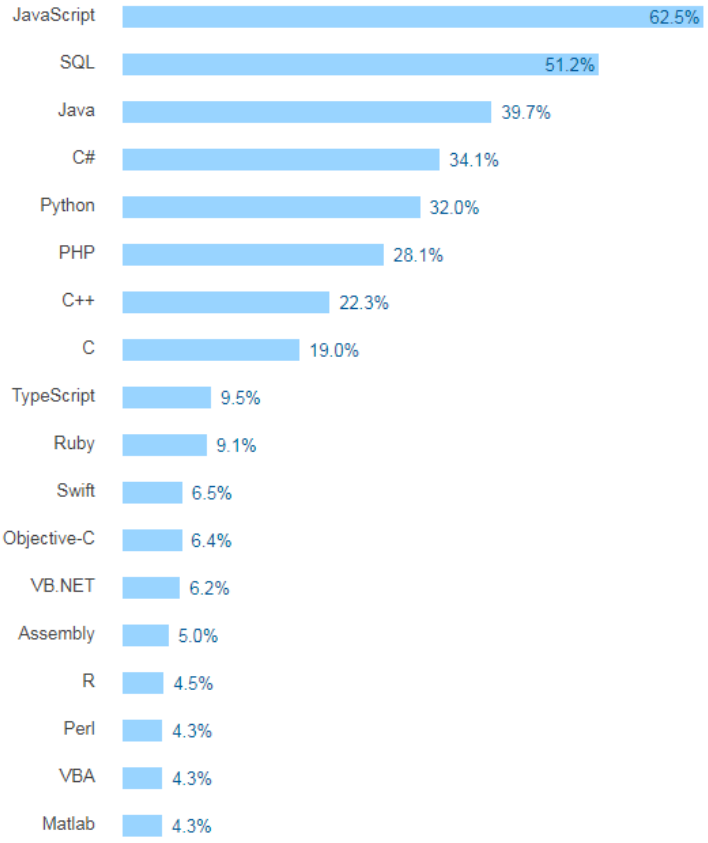

Рис. 12. Рейтинг от Stack Overflow

May 2018	May 2017	Change	Programming Language	Ratings	Change
1	1		Java	16.380%	+1.74%
2	2		C	14.000%	+7.00%
3	3		C++	7.668%	+2.92%
4	4		Python	5.192%	+1.64%
5	5		C#	4.402%	+0.95%
6	6		Visual Basic .NET	4.124%	+0.73%
7	9	^	PHP	3.321%	+0.63%
8	7	v	JavaScript	2.923%	-0.15%
9	-	^^	SQL	1.987%	+1.99%
10	11	^	Ruby	1.182%	-1.25%
11	14	^	R	1.180%	-1.01%
12	18	^^	Delphi/Object Pascal	1.012%	-1.03%
13	8	vv	Assembly language	0.998%	-1.86%
14	16	^	Go	0.970%	-1.11%

Рис. 13. Рейтинг от TIOBE

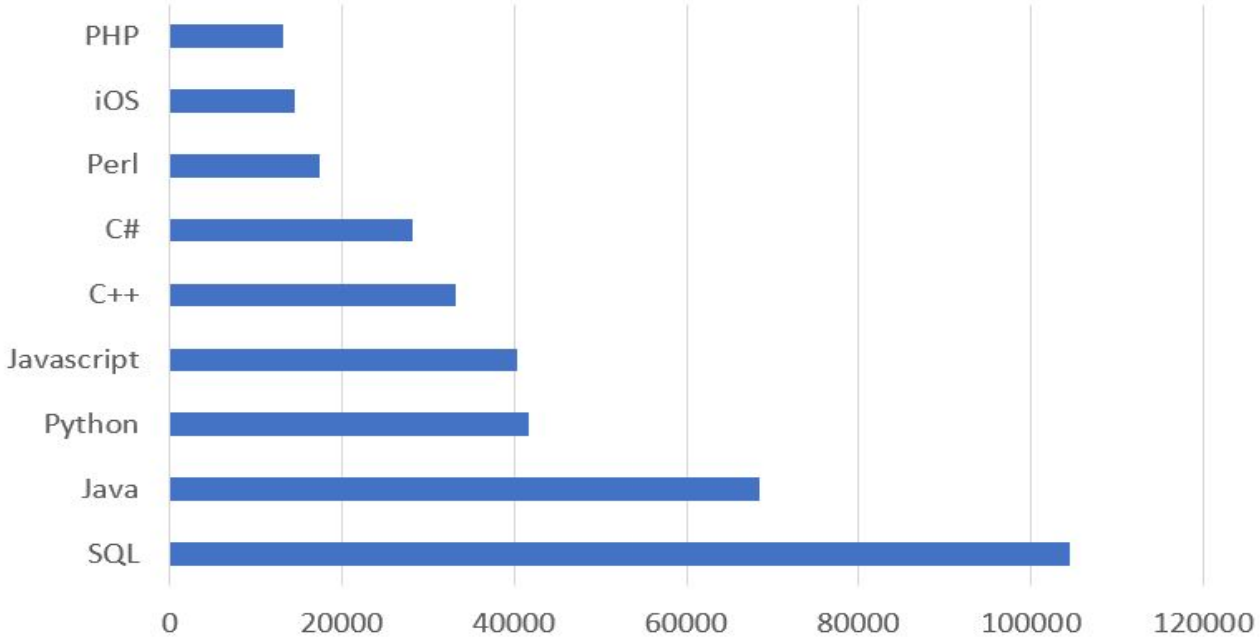

Рис. 14. Рейтинг от Coding Dojo по востребованности специалистов на рынке труда